PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-047894
(43)Date of publication of application : 18.02.2000

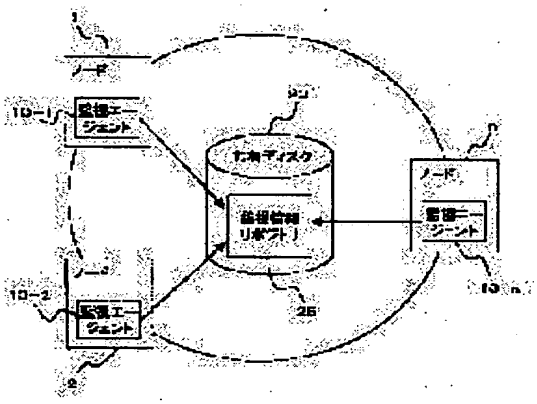
(51)Int.Cl. G06F 11/20
G06F 11/30

(21)Application number : 10-218384
(22)Date of filing : 31.07.1998

(71)Applicant : MITSUBISHI ELECTRIC CORP
(72)Inventor : KANO KEIJI

(54) COMPUTER SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To improve resistance to fault while securing processing efficiency in a computer system constituted by plural nodes.
SOLUTION: In plural networked nodes 1, 2,..., n, monitoring agents 10-1, 10-2,..., 10-n respectively operate to monitor the operating situation of a DB application in its own node and the other nodes. In addition, information of each node is stored in a monitoring information repository 25 on a shared disk 20 mutually shared by the respective nodes. This information includes the CPU load, the free memory quantity, etc., of each memory. A monitoring agent detecting the down of application in its own node and the down of the other nodes dynamically selects an alternative node by the reference of a small CPU load, e.g. based on the information of the repository 25 and instructs taking over of processing to this.



LEGAL STATUS

[Date of request for examination]	03.08.1998
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3062155
[Date of registration]	28.04.2000
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－47894
（P2000－47894A）
(43)公開日　平成12年２月18日(2000.2.18)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　11/20	３１０	Ｇ０６Ｆ　11/20	３１０Ｄ	５Ｂ０３４
11/30		11/30	Ｆ	５Ｂ０４２

審査請求　有　　請求項の数７　ＯＬ　（全 29 頁）

(21)出願番号　特願平10－218384

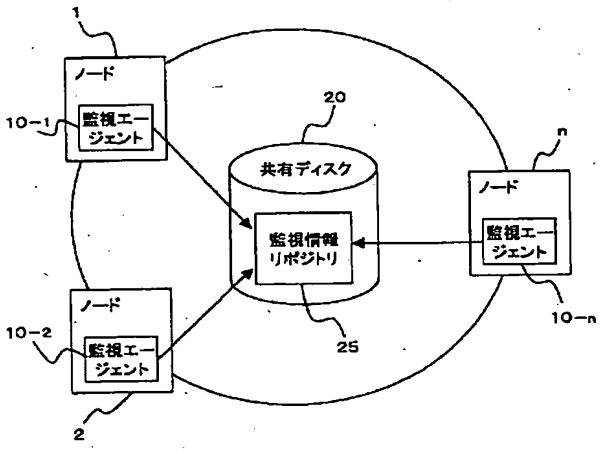
(22)出願日　平成10年７月31日(1998.7.31)

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号
(72)発明者　加納　敬次
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内
(74)代理人　100075258
弁理士　吉田　研二　（外２名）

Ｆターム(参考)　5B034 BB03 CC01 DD05
5B042 GA12 JJ04 JJ17

(54)【発明の名称】　計算機システム

(57)【要約】
【課題】　複数のノードで構成された計算機システムにおいて、処理効率を確保しつつ耐故障性を向上させる。
【解決手段】　ネットワーク接続された複数のノード１，２，…，ｎではそれぞれ監視エージェント１０-1，１０-2，…，１０-nが動作し、自ノード及び他ノードでのＤＢアプリケーションの稼働状況を監視する。また、各ノードで相互に共有する共有ディスク２０上の監視情報リポジトリ２５に各ノードの情報が格納される。この情報には、各ノードのＣＰＵ負荷、空きメモリ量などが含まれる。自ノードでのアプリケーションのダウン、他ノードのダウンを検知した監視エージェントは、監視情報リポジトリ２５の情報を基に、例えばＣＰＵ負荷が少ないといった基準で動的に代替ノードを選択して、これに処理の引き継ぎを指示する。



1

【特許請求の範囲】
【請求項1】　それぞれ処理プログラムを実行可能な複数のノードがネットワーク接続された計算機システムにおいて、
前記各ノードは、
前記各ノードそれぞれの稼働状況を監視する監視手段と、
前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき、代替ノードを選択する代替ノード選択手段と、
前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段と、
を有することを特徴とする計算機システム。
【請求項2】　請求項1記載の計算機システムにおいて、
前記監視手段は、前記各ノードそれぞれのプロセッサ負荷を監視し、
前記代替ノード選択手段は、前記監視手段により得られた前記プロセッサ負荷に基づいて前記代替ノードを選択すること、
を特徴とする計算機システム。
【請求項3】　請求項2記載の計算機システムにおいて、
前記代替ノード選択手段は、前記プロセッサ負荷が最小であるノードを前記代替ノードとして選択すること、
を特徴とする計算機システム。
【請求項4】　請求項1記載の計算機システムにおいて、
前記監視手段は、前記各ノードそれぞれのメモリ空き容量を監視し、
前記代替ノード選択手段は、前記監視手段により得られた前記メモリ空き容量に基づいて前記代替ノードを選択すること、
を特徴とする計算機システム。
【請求項5】　請求項2記載の計算機システムにおいて、
前記代替ノード選択手段は、前記メモリ空き容量が最大であるノードを前記代替ノードとして選択すること、
を特徴とする計算機システム。
【請求項6】　請求項1記載の計算機システムにおいて、
前記監視手段は、前記各ノードそれぞれのプロセッサ負荷とメモリ空き容量とを監視し、
前記代替ノード選択手段は、前記監視手段により得られた前記プロセッサ負荷と前記メモリ空き容量とに基づいて前記代替ノードを選択すること、
を特徴とする計算機システム。
【請求項7】　請求項1記載の計算機システムにおいて、

2

前記監視手段は、前記各ノードそれぞれのメモリ空き容量を監視し、
前記フェールオーバ手段は、
前記代替ノードの前記メモリ空き容量が前記処理プログラムの引き継ぎに十分か否かを判断する容量比較手段と、
前記メモリ空き容量が不足するときは、前記代替ノードで先行して起動されている先起動プログラムに割り当てられるメモリ容量を縮小するメモリ割当変更手段と、
を有することを特徴とする計算機システム。
【請求項8】　請求項7記載の計算機システムにおいて、
前記代替ノードの前記先起動プログラムと当該先起動プログラムを前記ネットワークを介して利用するユーザノードとの間のセッション情報を含んだセッション情報テーブルを有し、
前記フェールオーバ手段は、前記メモリ容量の割当変更を行う際に、前記セッション情報に基づいて既設セッション中のトランザクションが継続中か否かを判別して継続中のトランザクションに対するメッセージのみ前記先起動プログラムに渡し、当該セッションに対する他のメッセージは前記割当変更が完了するまで前記セッション情報に保留するメッセージ取扱手段を有し、
前記メモリ割当変更手段は、前記トランザクションが終了したときに、前記先起動プログラムの実行を停止してメモリ割当変更を行うこと、
を特徴とする計算機システム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、複数の計算機が接続された計算機システムに関し、特に耐故障性、可用性が向上した計算機システムに関する。
【０００２】
【従来の技術】計算機システムの可用性を向上させるためには、故障によるシステムのダウン時間を低減する必要がある。この耐故障性向上を図る一つの方法として冗長性の利用が挙げられる。冗長性はハードウェア、ソフトウェアの両方に適用することが可能であり、具体的には多重化、多数決方式、誤り訂正／検出符号の利用といった例がある。その中で、ハードウェアについて利用できる比較的簡単な形態であって適用範囲の広いものに、一つの計算機システムを互いに接続された複数の計算機で構成するという多重化の形態がある。
【０００３】この多重化では、システムを構成する個別の計算機（これを以下、ノードと呼ぶことがある）が何らかの原因で停止（ダウン）したとき、そのノードの処理を予め定義された別のノードが引き継ぐフェールオーバ構成とする事例が多くなってきている。
【０００４】
【発明が解決しようとする課題】この従来のフェールオ

ーバ構成では、障害発生時に、予め決められたノード先にしか処理を引き継ぐ（フェールオーバする）ことができない。つまり、この方法では、処理引き継ぎに必要な計算機リソース（資源）量（例えばＣＰＵ負荷、メモリ量）が多くても、それに応じてフェールオーバされるノードを動的に変更できず、フェールオーバされたノードの負荷が高くなってその処理速度が許容限度以上に低減したり、極端な場合には処理を引き継ぐことが実質的に不可能である場合が生じ得るという問題があった。

【０００５】また、例えば、重要度の低いデータベースアプリケーションソフトウェア（以下、ＤＢアプリケーションと略記する）が稼働しているノードに、重要度の高いＤＢアプリケーションをフェールオーバしたい場合がある。この場合、フェールオーバ先のノードのメモリの一部は、重要度の低いＤＢアプリケーションが先に占有しており、フェールオーバされる重要度の高いＤＢアプリケーションは、残りのメモリ資源しか利用することができず、処理が効率的に行われない、処理速度が遅いといった上述の不都合を生じる。このように従来のフェールオーバ方法では、単純には、リソースの割当が効果的に行われないといった問題があった。これを改善した従来のフェールオーバ方法には、一旦、先にリソースを占有している重要度の低いＤＢアプリケーションを停止させて、重要度の高いＤＢアプリケーションに対して優先的にリソースが与えられるような再配分を行うものもあった。この方法では、一旦、ノードをシャットダウンする、すなわち既存のセッション情報（端末とＤＢ間の接続及びトランザクションを処理するために必要な情報）が失われてしまい、先に実行されていたアプリケーションのリソース再配分後の実行が面倒となるという問題があった。

【０００６】本発明は上記問題点を解消するためになされたもので、複数ノードがネットワーク接続され、ノード間でフェールオーバを行うという多重化により耐故障性、可用性を向上させる計算機システムであって、フェールオーバ先での処理が効率的に行われるシステムを提供することを目的とする。

【０００７】

【課題を解決するための手段】本発明に係る、それぞれ処理プログラムを実行可能な複数のノードがネットワーク接続された計算機システムは、前記各ノードが、前記各ノードそれぞれの稼働状況を監視する監視手段と、前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき代替ノードを選択する代替ノード選択手段と、前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段とを有することを特徴とする。

【０００８】他の本発明に係る計算機システムにおいては、前記監視手段は前記各ノードそれぞれのプロセッサ負荷を監視し、前記代替ノード選択手段は前記監視手段により得られた前記プロセッサ負荷に基づいて前記代替ノードを選択することを特徴とする。本発明の好適な態様は、前記代替ノード選択手段が前記プロセッサ負荷が最小であるノードを前記代替ノードとして選択するものである。

【０００９】別の本発明に係る計算機システムにおいては、前記監視手段は前記各ノードそれぞれのメモリ空き容量を監視し、前記代替ノード選択手段は前記監視手段により得られた前記メモリ空き容量に基づいて前記代替ノードを選択することを特徴とする。本発明の好適な態様は、前記代替ノード選択手段が前記メモリ空き容量が最大であるノードを前記代替ノードとして選択するものである。

【００１０】また、本発明の他の好適な態様は、前記監視手段が前記各ノードそれぞれのプロセッサ負荷とメモリ空き容量とを監視し、前記代替ノード選択手段が前記監視手段により得られた前記プロセッサ負荷と前記メモリ空き容量とに基づいて前記代替ノードを選択するものである。

【００１１】また別の本発明に係る計算機システムにおいては、前記監視手段は前記各ノードそれぞれのメモリ空き容量を監視し、前記フェールオーバ手段は前記代替ノードの前記メモリ空き容量が前記処理プログラムの引き継ぎに十分か否かを判断する容量比較手段と、前記メモリ空き容量が不足するときは前記代替ノードで先行して起動されている先起動プログラムに割り当てられるメモリ容量を縮小するメモリ割当変更手段とを有することを特徴とする。

【００１２】さらに他の本発明に係る計算機システムは、前記代替ノードの前記先起動プログラムと当該先起動プログラムを前記ネットワークを介して利用するユーザノードとの間のセッション情報を含んだセッション情報テーブルを有し、前記フェールオーバ手段が、前記メモリ容量の割当変更を行う際に前記セッション情報に基づいて既設セッション中のトランザクションが継続中か否かを判別して継続中のトランザクションに対するメッセージのみ前記先起動プログラムに渡し、当該セッションに対する他のメッセージは前記割当変更が完了するまで前記セッション情報に保留するメッセージ取扱手段を有し、前記メモリ割当変更手段が前記トランザクションが終了したときに、前記先起動プログラムの実行を停止してメモリ割当変更を行うことを特徴とする。

【００１３】

【発明の実施の形態】［実施の形態１］以下、本発明の実施の形態について図面を参照して説明する。

【００１４】本発明に係る計算機システムは、複数のノード（ノード１、ノード２、…、ノードｎ）が互いにネットワークを介して接続されて構成される。図1は、本システムの構成を説明するための模式図である。各ノードではそれぞれ、監視エージェントと呼ぶモジュールが

5

動作する。この監視エージェントは、例えば、各ノードの中央処理装置(Central Processing Unit: ＣＰＵ)で実行されるソフトウェアで構成され、各ノードで常時実行状態にされている。そして監視エージェントは自律的に動作して、それが動作するノード(自ノードと称する) と自ノード以外のネットワーク接続されたノード(他ノードと称する) のＤＢアプリケーションの稼働状況、ＣＰＵ資源、メモリ資源などを定期的に、又は随時、サンプリングしながら監視する機能を有する。この監視エージェント１０は、ノードのオペレーティングシステム(ＯＳ) やＤＢが起動しているかどうかを判別する既存の機能を用いて各ノードの情報をチェックし、それぞれの監視エージェント１０-1, １０-2, …, １０-nの監視結果は、ネットワークを介して各ノードと接続された共有ディスク２０に書き込まれる。監視結果は具体的には、この共有ディスク２０上に設けられる監視情報リポジトリ２５に記録される。

【００１５】図2は、監視情報リポジトリ２５に格納される情報の一例の論理構成を示す模式図である。なお、この図において、データとして実際に値が取得されるのは、ツリー構造の最下位レベルの項目であり、それ以外の項目はその下位レベルを総括する単なる名称である。この図に示したように、監視情報リポジトリ２５には「ノード情報」として各ノード毎の情報が格納される。各「ノード情報」は、ノード名、最終サンプリング時刻、監視エージェント情報、ロック済みエージェント情報、サンプリング間隔、サンプリングデータ保存期間、ノード状態、基準空きメモリ量といったデータを含んでいる。

【００１６】また各「ノード情報」はさらに、「ＤＢ情報」、「資源情報」という名称で総括されるデータを含んでいる。ＤＢ情報は、各ノードにて使用されるＤＢシステムの種類に応じた数だけの「ＤＢ」エントリで構成され、各「ＤＢ」エントリはＤＢ名、実行優先度、サンプリング時刻、ＤＢ状態、起動情報といったデータを含んで構成される。

【００１７】「資源情報」は、「資源データ」エントリで構成され、各「資源データ」エントリはサンプリング時刻、ＣＰＵ稼働率、空きメモリ量といったデータを含んで構成される。「資源データ」エントリは「資源情報」内に、サンプリング保存期間に応じて定まる、結果を保存すべきサンプリング回数分だけ含まれる。すなわち、「資源情報」には、サンプリングが行われる度に、そのサンプリング時における「資源データ」を構成するデータセットが追加され、一方、予め定められたサンプリング保存期間を過ぎた「資源データ」エントリは削除される。

【００１８】以下、監視情報リポジトリ２５内の各データについて説明する。「ノード名」は、ノードを識別するための名前であり、「最終サンプリング時刻」は、当

6

該ノードについての情報が最後にサンプリングされた日付、時刻を表す情報である。「監視エージェント情報」は、ノードで動作する監視エージェントの識別子(名前) 、及び別の監視エージェントとのメッセージ交換ができるようにするための設定情報である。「ロック済みエージェント情報」は、当該ノードをチェックしている監視エージェントの識別子を表し、どの監視エージェントも当該ノードをチェック中でないときは値“ＮＵＬＬ”が設定される。

【００１９】「サンプリング間隔」は、監視エージェントが自ノード内の全てのＤＢ情報及び資源情報を採取し、監視情報リポジトリ２５内に登録されたＤＢアプリケーションがダウンしているかどうかをチェックする間隔である。「サンプリングデータ保存期間」には、資源情報の過去の履歴の保存期間が指定される。

【００２０】「ノード状態」は、現在のノードの状態を表すデータであり、その状態としては、例えば“停止中”、“稼働中”、“ダウン”、“フェールオーバ作業中”の4通りを定義することができる。

【００２１】ここで、“停止中”は、ノードが稼働していないことを示す状態である。“停止中”が記録されるタイミングはＯＳが正常に停止されるときに停止処理の一環として行われるか、ノードの“ダウン”状態下での全てのＤＢアプリケーションのフェールオーバ作業が完了(当該作業の成功又は不成功は問わない) した直後である。

【００２２】“稼働中”は、ノードが稼働していることを示す状態である。“稼働中”が記録されるタイミングは、自ノードの監視エージェントによって、自分自身の起動初期処理として、又は“フェールオーバ作業中”状態が終了した段階である。

【００２３】“ダウン”は、ノードが過去に“稼働中”の状態であったが、現時点ではノードが停止している状態を表す。“ダウン”が記録されるタイミングは、監視情報リポジトリに当該ノードの状態として“稼働中”が記録されているにも拘わらず他ノードの監視エージェントによって当該ノードの停止が検出された場合である。

【００２４】“フェールオーバ作業中”は、当該ノードにてフェールオーバが実際に行われている状態である。“フェールオーバ作業中”が記録されるタイミングは当該ノードの監視エージェントが他ノードの監視エージェントからＤＢフェールオーバの指示を受け取ったときであり、フェールオーバ作業が終了した時点で状態は“稼働中”に戻る。

【００２５】次に「基準空きメモリ量」には、起動直後の空きメモリ量が記録される。

【００２６】「ＤＢ情報」エントリに含まれる「ＤＢ名」は、当該エントリに対応するＤＢシステムを一意に特定する識別子である。「サンプリング時刻」は、当該「ＤＢ情報」エントリの情報がサンプリングされた日

付、時刻を表す。「起動情報」は当該ＤＢシステムに対応したＤＢアプリケーションを起動するための実行手順が記述されたスクリプトファイル名や、運用時の標準メモリ量など、ＤＢアプリケーションを起動するために必要な情報である。

【００２７】「ＤＢ状態」は、当該エントリに対応するＤＢシステムの状態を表すデータであり、その状態としては、例えば“停止中”、“稼働中”、“ダウン”の3通りを定義することができる。

【００２８】ここで、“停止中”は、ＤＢアプリケーションが停止している状態を示す。“停止中”が記録されるタイミングはＤＢアプリケーションが正常に停止されるときに停止処理の一環として行われるか、自ノードの監視エージェントによって、自分自身の起動初期処理としてＤＢアプリケーションの停止が検出されたときである。

【００２９】“稼働中”は、ＤＢアプリケーションが正常に動作していることを示す状態である。“稼働中”が記録されるタイミングは、当該ＤＢアプリケーションが動作するノードの監視エージェントによって、当該ＤＢアプリケーションの状態が“稼働中”以外の状態において、当該ＤＢアプリケーションが動作していることを検出したときである。

【００３０】“ダウン”は、ＤＢアプリケーションが過去に“稼働中”の状態であったが、現時点では停止している状態を表すものである。“ダウン”が記録されるタイミングは、監視情報リポジトリに当該ＤＢアプリケーションの状態として“稼働中”が記録されているにも拘わらず自ノードの監視エージェントによってＤＢアプリケーションの停止が検出されたとき、又は当該ノードの状態が他ノードの監視エージェントによって“ダウン”と記録される場合である。

【００３１】「実行優先度」は、ＤＢシステムの優先度であり、数値を以て表すことができる。例えば、数値が大きいほど優先度が高いと定義することができる。

【００３２】「資源情報」エントリに含まれる「サンプリング時刻」は、当該「資源情報」エントリの情報がサンプリングされた日付、時刻を表す。

【００３３】「ＣＰＵ稼働率」は、サンプリングが行われた時点でのＣＰＵの稼働率を表す。

【００３４】「空きメモリ量」は、サンプリングされた時刻での空きメモリ量を表す。

【００３５】図3～7は、監視エージェントの動作を説明するフロー図であり、特に図3は、監視エージェントのＤＢアプリケーション状態監視に関する初期処理を説明するフロー図である。まず、監視エージェント１０が起動される前においては、監視情報リポジトリ２５内のノード状態は“停止中”に設定される。各ノードが起動され、監視エージェント１０が起動されると、監視エージェント１０は自ノードの状態を“稼働中”に設定する（Ｓ５０）。次に、監視情報リポジトリ２５の自ノードに関するノード情報にエントリされているＤＢ情報に基づいて、エントリされている各ＤＢシステム毎に以下に説明するステップＳ６０～７５で構成されるループを繰り返す。このループに含まれる処理においては、まずＤＢの状態が監視情報リポジトリ２５から取り出される（Ｓ６０）。そして取り出したＤＢ状態の値が“稼働中”であるか否かが判定され（Ｓ６５）、その判定結果に応じた値でＤＢ状態が更新される（Ｓ７０，Ｓ７５）。なお、この初期処理においては、“ダウン”状態は生じないので、ステップＳ６５にて“稼働中”でないと判定された場合には、ステップＳ７５において、初期値“停止中”が維持される。ステップＳ８０においては、全てのＤＢシステムについてステップＳ６０～Ｓ７５が実施されたかが判断され、実施されていないＤＢシステムがあれば、処理はステップＳ６０に戻され、そのＤＢシステムについて同様の処理が行われる。一方、ステップＳ８０において全てのＤＢシステムについて処理が行われたと判断された場合には、図3に示す初期処理は終了し、各ノード毎で行われる監視処理が開始される。

【００３６】図4は、各ノードでの監視処理の概略を説明するフロー図であり、図5は当該監視処理に設けられる自ノードに対するＤＢ状況チェック処理Ｓ１２５内での処理をより詳細に示したフロー図である。図3と図4とは、図に示される端子Ａにおいて互いに接続される。図3に示す初期処理から図4に示す監視処理に処理が渡されると、以降、ノードが稼働中である間、監視処理を構成するＳ１００～１４５が例えば一定間隔で起動され繰り返される。

【００３７】タイマー処理Ｓ１００は、監視処理を一定間隔で起動するために設けられ、当該処理が所定時間の経過を検知したことに基づいて、続くＤＢ再起動処理Ｓ１０５が必要に応じて行われる。このＤＢ再起動処理Ｓ１０５は、ダウンしているＤＢアプリケーションの再起動先（フェールオーバ先）として自ノードが選択されたという通知を受けた場合に実行され、そのダウンしているＤＢアプリケーションを自ノードにおいて再起動する。

【００３８】次に時刻が取得され（Ｓ１１０）、その時刻に基づいて、監視情報リポジトリ２５内に格納されている自ノード情報内の資源情報のうちサンプリング保存期間を超えたものがあるかどうかが探索され、当該保存期間を超えたものは監視情報リポジトリ２５から削除される（Ｓ１１５）。

【００３９】このようにまず監視情報リポジトリ２５内の不要なデータを整理した後、監視エージェントは自ノードのチェックを開始する。そのチェック開始に際して、監視エージェントは自身の監視エージェント識別子を自ノード情報のロック済みエージェント情報に登録す

る（Ｓ１２０）。これにより自ノードに対するチェック処理実行中における他ノードの監視エージェントからのチェックを排他制御する。

【００４０】自ノードに対するチェック処理はＤＢアプリケーションの状況のチェックと、自ノードのリソース負荷状況のチェックとを含んでいる。このうち自ノードのＤＢアプリケーションに対するチェック処理Ｓ１２５を図5を用いてより詳しく説明する。このチェックは、監視情報リポジトリ２５の自ノード情報にエントリされている全てのＤＢシステムについて反復される（Ｓ２００）。まずＤＢ状態が監視情報リポジトリ２５から取り出され、当該ＤＢ状態の値が“稼働中”であるか否かが判定され（Ｓ２０５）、状態が“稼働中”でない場合には、当該ＤＢアプリケーションについてのチェックを終了し、残る未チェックのＤＢアプリケーションに対するチェックに移る（Ｓ２３０）。一方、ステップＳ２０５において、監視情報リポジトリ２５上に格納されているＤＢ状態データが“稼働中”であるＤＢシステムに対しては、そのＤＢシステムの実際の状態が“正常稼働”であるか“停止”であるかが調べられる（Ｓ２１０）。この実際の状態の判別には、従来技術を用いることができる。

【００４１】監視情報リポジトリ２５上で“稼働中”であった（Ｓ２０５）にも拘わらず、ステップＳ２１０で取得された実際の状態が“停止”であった場合（Ｓ２１５）は、ＤＢアプリケーションは実行障害を生じた、つまり“ダウン”していると判断され（Ｓ２２０）、ＤＢアプリケーションダウンに対応する所定の処理（Ｓ２２５）が実行される。

【００４２】このＤＢアプリケーションダウン時の処理では、実行障害を生じたＤＢアプリケーションを代替実行させるのに適当なノード（代替ノード）が選択され、当該ダウンしたＤＢアプリケーションをその代替ノードにおいて再起動し処理を引き継がせるための処理が、その代替ノードで動作している監視エージェントに対して実行される。すなわち、監視エージェントは、自ノードでのＤＢアプリケーションの実行状況を監視する監視手段、代替ノードを選択する代替ノード選択手段、及びダウンしたＤＢアプリケーションの実行を代替ノードへ引き継がせるフェールオーバ手段としての機能を有する。なお、ここで選択された代替ノードへ発せられる処理引き継ぎの指示は、当該代替ノードにおけるステップＳ１０５で検知され、実行される。

【００４３】さて一方、条件判断ステップＳ２１５において、”停止”ではない、すなわち実際に“稼働中”であることが明らかになった場合は、監視情報リポジトリ２５上の情報と一致しており異常はないので、当該ＤＢアプリケーションについてのチェックを終了し、残る未チェックのＤＢアプリケーションに対するチェックに移る（Ｓ２３０）。なお、ステップＳ２３０において、全

てのＤＢアプリケーションについてのチェックが完了したと判断された場合には、自ノードに対するチェック処理Ｓ１２５を終了する。

【００４４】チェック処理Ｓ１２５が終了すると、次に監視エージェントは、自ノードに関する資源情報のチェックを行う（Ｓ１３０）。この処理では資源情報データがＯＳから取得され、監視情報リポジトリ２５に格納される（Ｓ１３０）とともに、そのサンプリング時刻で監視情報リポジトリ２５内の自ノードに関する最終サンプリング時刻を更新する（Ｓ１３５）。

【００４５】ステップＳ１２５～１３５の自ノードに対するチェック処理が終了すると、監視エージェントは自ノード情報のロック済みエージェント情報に“ＮＵＬＬ”を登録し、排他制御を解除する（Ｓ１４０）。

【００４６】以上の処理は監視エージェントが自ノードに対して行う処理であるが、監視エージェントは自ノードだけでなく、他のノードに対する監視及び障害時のフェールオーバ処理をも行う（Ｓ１４５）。これを全ノード処理と称する。図6，７は、全ノード処理を説明するフロー図である。この全ノード処理は便宜上、図6，７の2つに分割して表しており、両図の同一符号の端子（具体的には端子B同士、及び端子C同士）において両図は接続される。この処理は、あるノードの監視エージェントが他のノードを順次選択して、選択したノードに関し稼働状況の監視、及び障害発生時のフェールオーバ処理を行うものであり、ステップＳ３００とステップＳ４４５とで挟まれる処理が、それぞれのノードに対して反復されるループ処理として構成されている。

【００４７】ステップＳ３０５～３３０は、あるノードの監視エージェント（以下、検査実行エージェントと称する）がチェック対象として選択したノード（以下、検査対象ノードと称する）を、他のノードの監視エージェントがロック状態としたままダウンした場合に対応する処置である。検査対象ノードのロック済みエージェント情報が判別され（Ｓ３０５）、それが“ＮＵＬＬ”である場合は問題がないので、検査実行エージェントは検査対象ノードのロック済みエージェント情報に、自身のノードの識別子を登録して検査対象ノードをロック状態とし（Ｓ３３５）、検査対象ノードに対するチェックを開始する。

【００４８】一方、ステップＳ３０５において、検査対象ノードのロック済みエージェント情報に、他ノードの識別子が登録されている場合は、当該他ノードの監視エージェントと通信可能かどうかが調べられる（Ｓ３１０）。もし通信可能であるならば、当該他ノードの監視エージェントがその検査対象ノードをチェックであると解して、当該他ノードに対するチェックは行わない。反対に、ステップＳ３１０において、通信不可能であることが判明した場合には、当該他ノードの監視エージェントは検査対象ノードをロックしたままダウンしたと解さ

れる。そこで、検査実行エージェントは、現在検査を行おうとしている検査対象ノードだけでなく全ノードを検索対象として、当該他ノードの監視エージェントによりロックされているノードを監視情報リポジトリ２５のノード情報に基づいて見いだし、そのロックを解除する。具体的には、全てのノードを対象とするループ処理が行われ（Ｓ３１５，Ｓ３３０）、そのノードのロック済みエージェント情報が当該他ノードの識別子であるかどうかが判定され（Ｓ３２０）、判定結果が“Ｙｅｓ”である場合にはそのロック済みエージェント情報を“ＮＵＬＬ”にリセットし（Ｓ３２５）、“Ｎｏ”である場合にはリセットを行わないという処理が行われる。しかる後に、検査実行エージェントは現在検査を行おうとしている検査対象ノードのロック済みエージェント情報に、自身のノードの識別子を登録して検査対象ノードをロック状態とし（Ｓ３３５）、検査対象ノードに対するチェックを開始する。

【００４９】検査実行エージェントは、まず、監視情報リポジトリ２５中の検査対象ノードのノード状態をチェックする（Ｓ４００）。このノード状態に“稼働中”、又は“フェールオーバ作業中”が設定されている場合は、さらに、検査対象ノードが実際に動作しているかどうかが調べられる（Ｓ４０５）。ちなみに、この判定は従来技術を用いて行うことができる。この判定の結果、検査対象ノードが、監視情報リポジトリ２５のノード状態に設定された情報に反して、実際は停止していることが判明した場合は、検査実行エージェントは検査対象ノードがダウンしていると判定して、監視情報リポジトリ２５の当該ノード状態を“ダウン”に変更する（Ｓ４１０）。そして、検査実行エージェントは、ダウンしていると判定した検査対象ノードについて監視情報リポジトリ２５に登録されている全てのＤＢ情報を対象とするループ処理を行う（Ｓ４１５，Ｓ４３０）。このループ処理では、当該ＤＢ情報中のＤＢ状態に“稼働中”が設定されているかどうかが調べられ（Ｓ４２０）、判定が“Ｙｅｓ”である場合にはＤＢ状態を“ダウン”に変更する（Ｓ４２５）。

【００５０】しかる後、検査実行エージェントは、ダウンしたと判定した検査対象ノード上で実行されていたＤＢアプリケーションに対し、ノードダウンに対応する所定の処理（Ｓ４３５）を実行する。このノードダウン時の処理では、当該ダウンノード上で実行されていたＤＢアプリケーションを代替実行させるのに適当なノード（代替ノード）が選択され、当該ＤＢアプリケーションをその代替ノードにおいて再起動し処理を引き継がせるための処理が、その代替ノードで動作している監視エージェントに対して実行される。すなわち、検査実行エージェントは、他ノードである検査対象ノードの稼働状況を監視する監視手段、代替ノードを選択する代替ノード選択手段、及びノードダウンにより実行できなくなった

ＤＢアプリケーションの実行を代替ノードへ引き継がせるフェールオーバ手段としての機能を有する。なお、ここで選択された代替ノードへ発せられる処理引き継ぎの指示は、当該代替ノードにおけるステップＳ１０５で検知され、実行される。ノードダウン時の処理Ｓ４３５が終了すると、検査実行エージェントは、検査対象ノードのロック済みエージェント情報を“ＮＵＬＬ”に変更しロックを解除し（Ｓ４４０）、新たな検査対象ノードに対する処理に移る（Ｓ４４５）。もし、全てのノードに対して処理が終了したら全ノード処理Ｓ１４５を終了し、図4に示すループに復帰する（Ｓ４５０）。

【００５１】なお、ステップＳ４００において、ノード状態が“停止中”である場合は問題がない、つまりＤＢアプリケーションの実行障害は生じていないので、直ちにロックを解除する（Ｓ４４０）。また、ノード状態が既に“ダウン”に設定されている場合は、他のノードの監視エージェントが既に当該検査対象ノードに対し検査を実行して“ダウン”を検知し、ノードダウン時の処理Ｓ４３５を実施したことを意味している。よって、この場合も、現在検査を行っている検査実行エージェントは改めてノードのダウンに対する処置を行わずに直ちにロックを解除する（Ｓ４４０）。

【００５２】上述したように本システムによれば、あるノードでのＤＢアプリケーションの実行が困難となったときに、ネットワーク接続された複数のノードの中から、その実行を引き継ぐノードとして適切なものが動的に選択される。

【００５３】［実施の形態２］本発明の第2の実施形態を、上記実施の形態1をベースに以下説明する。本実施の形態の特徴は、実施の形態1におけるＤＢアプリケーションダウン時の処理Ｓ２２５及びノードダウン時の処理Ｓ４３５に相当する処理にあり、以下、この実施の形態1との相違点に重きを置いて説明し、実施の形態1と基本的に同様である処理については説明を省略する。また、実施の形態1と同様の構成要素には同一の符号を付し、記載の簡潔を図る。

【００５４】本実施の形態においては、これらＤＢアプリケーションダウン時の処理及びノードダウン時の処理における代替ノードの選択が、代替ノードとして選択されるノードのＣＰＵ負荷に基づいて行われる点に大きな特徴がある。

【００５５】本システムの各ノード上の監視エージェント１０は、互いに非同期にサンプリング間隔毎に動作している。監視エージェント１０は、自ノード内のあるＤＢアプリケーションがダウンしていること、又は他のあるノード全体がダウンしていることを検出すると、実施の形態1のステップＳ２２５又はＳ４３５に相当するＤＢアプリケーションダウン時の処理及びノードダウン時の処理を行う。

【００５６】図8は、本実施の形態におけるＤＢアプリ

13

ケーションダウン時の処理を示すフロー図である。この処理は、ノード自体は稼働しているが、DBアプリケーションはダウンしていることが検知された場合に実施される処理であり、図4のステップS120に続いて行われる。

【0057】監視エージェント10は、自ノードのDBアプリケーションにダウン状態のものを検知すると、監視情報リポジトリ25に登録されている全てのノードを対象とした以下の内容のループ処理を行う（S500，S525）。このループ処理の1回のループは1つのノードに対応した処理であり、まず、そのノードの状態が“稼働中”、“フェールオーバ作業中”のいずれかであるかどうかが監視情報リポジトリ25のノード情報に基づいて判定される（S505）。判定結果が“Yes”である場合には、当該ノードに関する監視情報リポジトリ25中の資源情報に基づいて、平均CPU稼働率が算出される（S510）。ちなみに、例えばこの平均稼働率は、監視情報リポジトリ25に保存されている当該ノードの複数サンプリングタイミングにおける資源情報の全てを用いて計算される。

【0058】次に、当該ノードが、現在の検査実行エージェント以外の他ノードの監視エージェントにより“フェールオーバ作業中”であるか否かが判定され（S515）、その結果に基づいて換算CPU稼働率が算定される。換算CPU稼働率は、ステップS510で計算された平均CPU稼働率に換算率αを乗じることにより求められる。ステップS510にてフェールオーバ作業中と判定された場合には、αは1以上の定数（例えば10）とされ、これが乗じられて換算CPU稼働率が求められる（S520）。一方、ステップS515にて、フェールオーバ作業中ではないと判定された場合には、αは1とされる、すなわち、ステップS510で求められた平均CPU稼働率がそのまま換算CPU稼働率とされる。監視エージェント10は、この換算CPU稼働率を決定すると、当該ノードについてのループ処理を終了し、次のノードのループ処理に移る。

【0059】なお、ステップS505での判定結果が、“停止中”、又は“ダウン”である場合には、当該ノードにDBアプリケーションの処理を引き継ぐことはできないので、代替ノードの候補から外して、次のノードに対するループに移る。例えば、代替ノードの候補から外したことは、それを表すためのフラグ等を設け、それにより識別することができる。また、本システムでは後述するように換算CPU稼働率が小さいことを基準として代替ノードを選択するので、ノードの換算CPU稼働率に非常に大きな値を設定することにより、当該ノードが代替ノード候補外であることを識別するように構成することもできる。

【0060】監視エージェント10は上記ループ処理により、ネットワーク接続されたシステムの各ノードの代

14

替ノード候補としての可否、及び換算CPU稼働率を定めると、代替ノード候補のうち換算CPU稼働率が最も小さいノードを代替ノードとして選択し、当該代替ノードに対し、DBアプリケーションのダウンが確認されたノード名（すなわち自ノード名）とダウン状態のDB名をメッセージとして送信することによりダウンしたDBアプリケーションの再起動を指示し（S530）、図4のステップS130に復帰する。

【0061】ちなみに、ノード状態がフェールオーバ作業中である場合に1以上のαを乗じるのは、既にフェールオーバ作業中のノードへは新たなフェールオーバ作業の依頼が行われることを抑制し、なるべくフェールオーバ作業を行っていないノードへ依頼が行われるようにして処理分散を図るためである。

【0062】図9は、本実施の形態におけるノードダウン時の処理を示すフロー図である。この処理は、あるノードの監視エージェント10が他ノードの全体がダウンしていることを検知した場合に実施する処理であり、図7のステップS430に続いて行われる。

【0063】監視エージェント10は、監視情報リポジトリ25に登録されている全てのノードを対象とした以下の内容のループ処理を行う（S600，S650）。このループ処理の1回のループは1つのノードに対応した処理であり、そのノードの状態が“ダウン”であるかどうかが監視情報リポジトリ25のノード情報に基づいて判定される（S605）。判定結果が“No”である場合には、再起動等の実行障害処理は不要であるので、次のノードについてのループ処理に移る。一方、判定結果が“Yes”である場合には、ステップS610～S635の処理を行う。このステップS610～S635の処理は、代替ノード候補としての可否、及び換算CPU稼働率を定めるための処理であり、上に説明したステップS500～S525の処理と同様であるので、説明を省略する。

【0064】各ノードの換算CPU稼働率が決定されると、監視エージェント10は代替ノード候補のうち換算CPU稼働率が最も小さいノードを代替ノードとして選択し、当該代替ノードに対し、DBアプリケーションのダウンが確認されたノード名（この場合は、DBアプリケーションダウン時の処理と異なり、他ノード名となる）とダウン状態のDB名をメッセージとして送信する（S640）。さらに、ダウンと判定されたノードの監視情報リポジトリ25での状態を“停止中”に変更し（S645）、次のノードについてのループ処理に移る。このようにして、ノードダウン時の処理のループS600～S650は全てのノードを対象として実施される。

【0065】図10は、ステップS530，S640で送信されたメッセージを受信したノードでのDBアプリケーション再起動処理を説明するフロー図である。この

15

再起動処理は、図4のステップS105に相当する。

【0066】各ノードは、一定間隔で監視エージェント10が動作し、いずれかの監視エージェント10からのメッセージが存在するか否かを調べる（S700）。メッセージが存在しなければ、図4に示すステップS110以降の検査処理に移るが、メッセージが存在した場合は、以下に述べるDBアプリケーション再起動処理を行った後、検査処理に移る。

【0067】再起動処理では、メッセージからDBアプリケーションの実行障害が検知されたノード名とDB名が取得される（S705）。そして、監視情報リポジトリ25の時ノードの情報に、“フェールオーバ作業中”を設定し（S710）、監視情報リポジトリ25からステップS705で取得されたDB名に対応するDB起動情報を探しだし、その情報に基づいてDBアプリケーションを起動する（S715）。また、ステップS705で取得されたノード名、DB名に関するDB情報エントリを、自ノードのDB情報に移動させるとともに（S720）、当該DB情報の状態を“稼働中”に変更する（S725）。そして、この再起動処理を行った後に既に述べたように検査処理に移る。

【0068】図11は、上述した、DBアプリケーションの実行障害検出時の監視エージェント10の動作の一例を説明する模式図である。この図に示す例では、ノード1がダウンしていること、そしてその結果、ノード1で実行されていたDBアプリケーション「DB#1」がダウン状態であることが、ノード2の監視エージェント10（検査実行エージェント）による検査処理により検知される。検査実行エージェントは、監視情報リポジトリ25の各ノードのノード情報中のCPU稼働率を参照して、換算CPU稼働率を算出し、その最小値を有するノード3を代替ノードに選択する。そして、ノード3の監視エージェント10に対し、ノード1で実行されていたDB#1の処理を引き継ぐように指示する。この指示を受けたノード3の監視エージェント10は、監視情報リポジトリ25中のDB#1に関する起動情報等に基づいてその再起動を実行する。

【0069】本システムによれば、CPUの例えば過去の稼働状況に基づいて、CPU負荷が比較的厳しくないノードに、ダウンしたDBアプリケーションの処理引き継ぎが行われる。

【0070】なお、上述の例では、既にフェールオーバ作業中であるノードの負荷を考慮した換算CPU稼働率が最小となるノードを代替ノードに選択したが、そのような考慮を行わずに、平均CPU稼働率が最小値をとるノードを選択することとしてもよい。また、必ずしもそれらCPU稼働率が最も小さくなくてもよく、例えばCPU負荷が所定の余裕を有することに基づいて選択を行うことができ、この場合、例えば、CPU稼働率が所定の閾値以下となるノードのうち任意のものを選択することができる。

16

【0071】また、上述の例では、サンプリングデータ保存期間全体での平均に基づいて換算CPU稼働率、又は平均CPU稼働率を求めたが、例えば、最新の何回かのサンプリングで得られた資源情報の平均に基づいて代替ノードを選択することもできる。

【0072】さらに、資源情報に登録されるCPU稼働率についての平均以外の統計情報を求め、それを考慮して代替ノードを決定することもできる。例えば、CPU稼働率の分散等から推定されるCPU稼働率の時間的な変動を考慮して、単に平均値が小さいだけでなく、安定してCPU稼働率が小さいものを代替ノードに選択してもよい。

【0073】［実施の形態3］本発明の第3の実施形態を、図を参照して説明する。実施の形態2は、監視情報リポジトリ25に登録されるCPU稼働率を利用し、プロセッサ負荷に基づいて代替ノードを選択した。これに対し本実施の形態は、監視情報リポジトリ25に登録される空きメモリ量を利用し、それに基づいて代替ノードを選択する点が実施の形態2と異なる。以下、上記各実施の形態と同様の内容については説明を省略し、それらとの相違点に重きを置いて説明する。なお、上記各実施の形態と同様の構成要素には同一の符号を付し、記載の簡潔を図る。

【0074】図12は、本実施の形態の特徴的処理を説明するフロー図である。この処理は、図8の処理ステップS500～S530及び図9のループ処理ステップS610～S640に置き換わるものである。

【0075】すなわち、本システムの各ノード上の監視エージェント10は互いに非同期にサンプリング間隔毎に動作し、自ノード内のあるDBアプリケーションがダウンしていること、又は他のあるノード全体がダウンしていることを検出すると、実施の形態1のステップS225又はS435に相当するDBアプリケーションダウン時の処理及びノードダウン時の処理を行うわけであるが、本実施の形態におけるこれらの処理は、図12のフローで一部を置換された図8，9のフローで表される。

【0076】各ノードの監視エージェント10は、各ノードの基準空きメモリ量を監視情報リポジトリ25から取得する。基準空きメモリ量は、各ノードの起動直後の空き実メモリ量であり、基本的にはOS等のノード稼働に必須のソフトウェアのみメモリ上に存在し、他のアプリケーションがまだメモリ上に存在しない状態での空きメモリ量を表す。各監視エージェント10は、これら各ノードの基準空きメモリ量を比較して、その最大値M_{max}を求める。この処理は、例えば、図3に示すステップS50の処理中に行うように構成される。

【0077】次に、図12の処理フローを説明する。監視エージェント10は、自ノードのDBアプリケーションにダウン状態のものを検知した場合、及び他のノード

全体がダウンしていることを検知した場合に、監視情報リポジトリ２５に登録されている全てのノードを対象とした以下の内容のループ処理を行う（Ｓ８００，Ｓ８２０）。このループ処理の1回のループは1つのノードに対応した処理であり、監視エージェント１０はそのノードに関する資源情報に含まれる空きメモリ量を例えば監視情報リポジトリ２５に保存されている全サンプリング回数について平均し、その平均値をM_{max}で除した値として定義されるメモリ余裕度mを求める（Ｓ８０５）。

【００７８】続いて、監視エージェント１０は、当該ノードが、現在の検査実行エージェント以外の他ノードの監視エージェントにより“フェールオーバ作業中”であるか否かを判定し（Ｓ８１０）、その結果に基づいて換算メモリ余裕度が算定される。換算メモリ余裕度m^*は、ステップＳ８０５で計算されたメモリ余裕度mに換算率βを乗じることにより求められる。ステップＳ８１０にてフェールオーバ作業中と判定された場合には、βは0以上1未満の定数（例えば0．１）とされ、これを乗じて換算メモリ余裕度m^*が求められる（Ｓ８１５）。一方、ステップＳ８１０にて、フェールオーバ作業中ではないと判定された場合には、βは1とされる、すなわち、ステップＳ８１０で求められたメモリ余裕度mがそのまま換算メモリ余裕度m^*とされる。監視エージェント１０は、この換算メモリ余裕度m^*を決定すると、当該ノードについてのループ処理を終了し、次のノードのループ処理に移る。

【００７９】監視エージェント１０は上記ループ処理により、ネットワーク接続されたシステムの各ノードの代替ノード候補としての可否、及び換算メモリ余裕度を定めると、代替ノード候補のうち換算メモリ余裕度が最も大きいノードを代替ノードとして選択し、当該代替ノードに対し、ＤＢアプリケーションのダウンが確認されたノード名（すなわち自ノード名）とダウン状態のＤＢ名をメッセージとして送信することによりダウンしたＤＢアプリケーションの再起動を指示する（Ｓ８２５）。

【００８０】ちなみに、ノード状態がフェールオーバ作業中である場合に0以上1未満のβを乗じるのは、既にフェールオーバ作業中のノードへは新たなフェールオーバ作業の依頼が行われることを抑制し、なるべくフェールオーバ作業を行っていないノードへ依頼が行われるようにして処理分散を図るためである。

【００８１】本システムによれば、メモリの例えば過去の空き状況に基づいて、メモリの余裕が比較的大きいノードに、ダウンしたＤＢアプリケーションの処理引き継ぎが行われる。メモリの余裕が大きい場合には、スワッピングが抑制されることによりキャッシュのヒット率が向上したり、スラッシングが防止されることにより高い処理効率が得られる。

【００８２】なお、上述の例では、既にフェールオーバ作業中であるノードの負荷を考慮した換算メモリ余裕度

が最大となるノードを代替ノードに選択したが、そのような考慮を行わずに、単純なメモリ余裕度が最大値をとるノードを選択することとしてもよい。また、必ずしもそれらメモリ余裕度が最も大きくなくてもよく、例えばメモリ余裕度が所定の閾値以上となるノードのうち任意のものを選択することができる。

【００８３】また、上述の例では、サンプリングデータ保存期間全体での平均に基づいて換算メモリ余裕度、又は単純なメモリ余裕度を求めたが、例えば、最新の何回かのサンプリングで得られた資源情報の平均に基づいて代替ノードを選択することもできる。

【００８４】さらに、資源情報に登録される空きメモリ量（又はメモリ余裕度）についての平均以外の統計情報を求め、それを考慮して代替ノードを決定することもできる。例えば、メモリ余裕度の分散等から推定される空きメモリ量の時間的な変動を考慮して、単に平均値が大きいだけでなく、安定してメモリ余裕度が大きいものを代替ノードに選択してもよい。

【００８５】［実施の形態４］本発明の第4の実施形態は、監視情報リポジトリ２５に登録されるＣＰＵ稼働率と空きメモリ量との双方を利用して代替ノードを選択する点が、ＣＰＵ稼働率のみ利用する実施の形態2や空きメモリ量のみ利用する実施の形態3と異なる。以下、上記各実施の形態と同様の内容については説明を省略し、それらとの相違点に重きを置いて説明する。なお、上記各実施の形態と同様の構成要素には同一の符号を付し、記載の簡潔を図る。

【００８６】本実施の形態では、ＣＰＵ稼働率と空きメモリ量との双方を用いて「空き資源率」なるパラメータをノードごとに算出し、その値に基づいて代替ノードを選択する。つまり、一般に計算機における「資源」はＣＰＵ負荷、メモリ使用のそれぞれに関する独立した指標で把握されるが、本実施の形態では、それらの指標を、あるノードの資源のうち他の処理に利用可能な資源の割合を表す「空き資源率」という一つの指標に統合する。

【００８７】例えば、空き資源率は、次のように定義される。ＣＰＵ稼働率をW（$0 \leqq W \leqq 1$）で表すと、ＣＰＵ空き率ωは（１－W）で表される。メモリに関しては、空きメモリ量をＥで表し、また、実施の形態3と同様にしてM_{max}を求め、これらの比（Ｅ／M_{max}）をメモリ空き率εと定義する。そして、これらＣＰＵ空き率ωとメモリ空き率εとの相加平均{（$\omega+\varepsilon$）／2}を空き資源率と定義する。

【００８８】監視エージェント１０（検査実行エージェント）は、このように定義した空き資源率が最大となるノードを代替ノードに選択する。

【００８９】本発明の趣旨は、ＣＰＵ稼働率と空きメモリ量との双方を考慮して代替ノードを決定する点にあり、上に示した空き資源率の定義は一例に過ぎない。例えば、（$k_1 \cdot \omega + k_2 \cdot \varepsilon$）／（$k_1 + k_2$）、（$\omega^2 +$

19

ε^2) /2、($\omega \cdot \varepsilon$) $^{1/2}$といったもので空き資源率を定義することも可能である。

【００９０】このようにＣＰＵ稼働率と空きメモリ量との双方を考慮することにより、ノードの資源の余裕度の評価の信頼性が高くなり、より適切に代替ノードを決定することができる。

【００９１】[実施の形態5] 上記各実施の形態では、フェールオーバ先の空き資源の度合いを計算し、余裕があるノードへダウンしたＤＢアプリケーションの処理の引き継ぎを行わせるものであった。これらの方式では、代替ノードは動的に決定され、余裕度のあるものが選択されるので、そのようにして決定された代替ノードでフェールオーバが不可能である可能性は低い。しかし、それでもなお、資源が最も空いたノードを選択したにも拘わらずフェールオーバするには資源(特にメモリ資源) が足りない可能性は残る。

【００９２】本発明の第5の実施形態は、これに対処するものであり、フェールオーバ先のノードで現在稼働中のＤＢアプリケーションが獲得しているメモリを動的に変更、すなわち空きメモリ量を必要に応じて増加させ、フェールオーバを支障なく実行するものである。

【００９３】以下、上記各実施の形態と同様の内容については説明を省略し、それらと本実施の形態との相違点に重きを置いて説明する。なお、上記各実施の形態と同様の構成要素には同一の符号を付し、記載の簡潔を図る。

【００９４】図13は、本実施の形態の特徴的処理を説明するフロー図である。この処理は、基本的には図4のＤＢ再起動処理Ｓ１０５内に追加されるものである。

【００９５】すなわち、タイマー処理が監視エージェント１０の処理を開始させると、ＤＢ再起動処理Ｓ１０５が開始され、その中でまず、監視エージェント１０は、ＤＢ再起動を指示するメッセージを受信していることを確認すると、それに指示されたＤＢ名に対応する起動情報を監視情報リポジトリ２５から取得するとともに、現在の自ノードの空きメモリ量を把握する。そして起動情報中のデータである当該ＤＢシステムの運用に用いられる標準メモリ量(所要空きメモリ量) と、現在の空きメモリ量とを比較する(Ｓ９００)。もし、空きメモリ量が標準メモリ量より大きい場合には(Ｓ９００)、ＤＢ再起動に支障がないので、上記各実施の形態と同様のフェールオーバ処理によるＤＢの起動が開始される(Ｓ９２５)。

【００９６】一方、空きメモリ量が標準メモリ量より小さい場合には、そのままＤＢ再起動を行うと、ＤＢアプリケーションを起動できないか、起動できても運用に支障を生じる可能性がある。よって、この場合には監視エージェント１０は、監視情報リポジトリ２５の自ノード情報に登録されている先に起動されている全てのＤＢエントリのうち、フェールオーバされるＤＢアプリケーションより優先度の低いＤＢアプリケーションを対象とした以下の内容のループ処理を行う(Ｓ９０５，Ｓ９２０)。このループ処理の1回のループは1つのＤＢエントリに対応した処理であり、監視エージェント１０はそのＤＢエントリに対して、現在よりも縮小された新たなメモリ割り当て量を決定する(Ｓ９１０)。そして、当該先に起動されているＤＢアプリケーションを一旦終了した後、ステップＳ９１０で決定した割り当て量に基づいて再起動する(Ｓ９１５)。これによりメモリの再編成が行われ、ループ処理が対象とする全てのＤＢエントリについて処理が終わると、メモリ上にはフェールオーバ対象のＤＢアプリケーションの運用に必要な量の空きが生成されている。監視エージェント１０は、ステップＳ９００にて空きメモリ量が標準メモリ量より小さいと判定された場合には、このようにして空きメモリ量を拡大した後、上記各実施の形態で行ったと同様にしてフェールオーバ対象のＤＢアプリケーションを起動する(Ｓ９２５)。ちなみに、このステップＳ９２５におけるＤＢアプリケーションの起動は、監視情報リポジトリ２５に登録された起動情報中の起動のためのスクリプトファイル等を参照して実行される。

20

【００９７】先起動のＤＢアプリケーションに対する新たなメモリ割り当て量の具体的な決定方法の例を次に説明する。例えば、新たなメモリ割り当て量の決定においては、対象となるＤＢエントリの実行優先度を考慮することができる。監視情報リポジトリ２５には、各ＤＢエントリの実行優先度が登録されており、監視情報リポジトリ２５の説明で述べたように、ここでは、実行優先度はそれが高い程、大きな数値で表される。ここで、ＤＢエントリ「ＤＢ#i」の優先度を表す数値をP_iとすると、ＤＢエントリ「ＤＢ#a」に対する割り当てメモリの縮小量R_aを次式で計算する。なお、この式の中でLは、フェールオーバ対象ＤＢの起動に不足しているメモリ量であり、その標準メモリ量から現在の空きメモリ量を差し引いた値である。また、総和“Σ”はステップＳ９０５で対象とされる全てのＤＢエントリ(すなわち、フェールオーバ対象のＤＢエントリより優先度の低いもの) について計算される。

【００９８】$R_a = L \cdot P_a / \Sigma P_i$

そして、ＤＢエントリ「ＤＢ#a」に対する新たなメモリ割り当て量は、当該エントリに対して監視情報リポジトリ２５に格納されている運用時標準メモリ量からR_aを差し引いた値に決定される。

【００９９】このように、フェールオーバ対象のＤＢアプリケーションの実行に空きメモリ量が不足している場合でも、先行して起動されている他のＤＢアプリケーションに割り当てられるメモリ量を縮小・再編成することにより、フェールオーバ処理が一層、確実に実行される。

【０１００】[実施の形態6] 上記実施の形態5では、

21

先行して起動されているＤＢアプリケーションの使用メモリ量を変更することにより、フェールオーバ対象のＤＢアプリケーションの実行に必要な空きメモリ量が確保された。このメモリ割り当ての変更に際し、先行して起動されているＤＢアプリケーションは一旦終了される。本実施の形態は、この先行起動ＤＢアプリケーションの終了及び再起動の際のマンマシンインターフェースの改善に関わるものである。すなわち、本実施の形態は、稼働中のＤＢシステムのメモリ使用量を縮小してフェールオーバ対象のＤＢシステム起動に必要な空きメモリ量を確保する実施の形態5において、当該メモリ縮小時に、稼働中のＤＢシステムの処理の継続性を担保し、エンドユーザにＤＢシステムが停止されることを意識させないように構成したことを特徴とするものである。

【０１０１】以下、上記各実施の形態と同様の内容については説明を省略し、それらと本実施の形態との相違点に重きを置いて説明する。なお、上記各実施の形態と同様の構成要素には同一の符号を付し、記載の簡潔を図る。

【０１０２】エンドユーザとＤＢシステムとがネットワークを介してメッセージを交換しながら処理を進めるシステム形態においては、エンドユーザ側ノードとＤＢシステム動作ノードそれぞれの処理は複数の機能レイヤを含む階層構造に構成される。

【０１０３】図１４は、エンドユーザ側ノードとＤＢシステム動作ノードとの機能レイヤ構成例を示す模式図である。エンドユーザ側ノード１０００には、上位のレイヤからエンドユーザアプリケーション１００２、ＤＢシステム用の端末側メッセージ交換モジュール１００４及びネットワーク制御層１００６の3つのレイヤが示され、一方、ＤＢシステム動作ノード１０１０には、上位のレイヤからＤＢアプリケーション１０１２、ＤＢシステム用のサーバ側メッセージ交換モジュール１０１４及びネットワーク制御層１０１６の3つのレイヤが示されている。そして、両ノードは最下位のレイヤであるネットワーク制御層間を物理的なコネクション１０２０で接続される。なお、このレイヤ構成は一例であって、例えば各レイヤをさらに細かく分けることも可能である。例えば、ネットワーク制御層１００６，１０１６は、ＴＣＰ／ＩＰプロトコルではネットワーク層、データリンク層及び物理層に対応することとなる。

【０１０４】エンドユーザアプリケーション１００２から発行されるメッセージは、端末側メッセージ交換モジュール１００４を介して、ネットワーク制御層１００６に渡される。ネットワーク制御層１００６とネットワーク制御層１０１６との間では、ケーブルなどの物理的なコネクション１０２０を介して、信号の授受が行われる。ＤＢシステム動作ノード１０１０のサーバ側メッセージ交換モジュール１０１４は、ネットワーク制御層１０１６からのメッセージを常時受け入れることができるように監視を行い、メッセージの到着を確認すると、そのメッセージをＤＢアプリケーション１０１２に渡す。逆に、ＤＢアプリケーション１０１２からの結果出力は、サーバ側メッセージ交換モジュール１０１４、ネットワーク制御層１０１６、コネクション１０２０、ネットワーク制御層１００６、端末側メッセージ交換モジュール１００４を順にたどって、エンドユーザアプリケーション１００２に渡される。

22

【０１０５】さて、図１４に示すようなシステム構成において、フェールオーバ時には先行して稼働されている１又は複数のＤＢアプリケーション１０１２の実行プロセス（以下、ＤＢプロセスと呼ぶ）に対するメモリ量の変更処理が行われうる。このメモリ変更処理は、ＤＢこのとき、ＤＢプロセスを正常に終了させ、新たなメモリ量で再実行させることで実現され、その際のエンドユーザアプリケーション１００２と各ＤＢアプリケーション（ＤＢプロセス）１０１２との間のセッションの維持の機能は主としてサーバ側メッセージ交換モジュール１０１４が担う。すなわち、サーバ側メッセージ交換モジュール１０１４は、エンドユーザから見て、セッション（エンドユーザアプリケーションとＤＢアプリケーションのレベルで相互にメッセージの交換が可能な状態）があたかも切断されていない、つまり擬似的にセッションがつながっているように見せる機能を有している。

【０１０６】図１５は、本実施の形態の特徴的処理を含んだメモリ再編成処理を説明するフロー図である。この処理は、基本的には図１３のメモリ再編成処理Ｓ９１５の位置において実行される処理ステップを構成する。

【０１０７】監視エージェント１０は自ノードにおいて先行して起動されているＤＢプロセスに対し決定される新メモリ割り当て量を得ると（Ｓ１１００）、サーバ側メッセージ交換モジュールに対し、ＤＢ再起動準備要求信号を発する（Ｓ１１０５）。

【０１０８】この要求信号を受信したサーバ側メッセージ交換モジュールは、セッション維持のためのセッション処理（詳細は後述する）を行い、監視エージェント１０へ準備済み信号を送信する（Ｓ１１１０）。監視エージェント１０は、この準備済み信号を受信した後、チェックポイント処理を行う（Ｓ１１１５）。チェックポイント処理は、ＤＢプロセスのメモリ空間にあるデータベースのデータ断片と実際にディスクに記録されているデータ断片の内容とが異なる場合に、メモリ上のデータ断片をディスクに書き込む処理である。チェックポイント処理Ｓ１１１５を行った後に、監視エージェント１０はＤＢシステムを停止させる（Ｓ１１２０）。なお、その停止方法は、個々のＤＢシステムに依存する。

【０１０９】次に、監視エージェント１０は、各ＤＢプロセス毎に、磁気ディスク装置等に格納されている全てのＤＢパラメータファイルを読み出すループ処理を行う（Ｓ１１２５，Ｓ１１４０）。このループ処理内では、

磁気ディスク装置１１３０から例えば一つずつパラメータファイルを読み込む。このパラメータファイルは、ＤＢプロセスに割り当てるメモリ量等のＤＢプロセス起動条件を外部から指定するためのファイルであり、ＤＢシステムはこのパラメータファイルに応じた種々の条件で起動される。なお、その起動の仕方は個々のＤＢシステムに依存し様々である。さて、監視エージェント１０は磁気ディスク装置１１３０から読み込んだパラメータファイルからＤＢプロセスに割り当てられるメモリ量情報が取り出される（Ｓ１１３５）。

【０１１０】このようにループＳ１１２５～Ｓ１１３５を終了した段階では、あるＤＢシステムに対する種々の起動条件におけるメモリ所要量が取り出される。監視エージェント１０は、それらの中に、ステップＳ１１００で得た新メモリ割り当て量に等しいものを見いだした場合には、当該メモリ所要量に対応するパラメータファイルを、再起動に用いるパラメータファイルとして選択する。また、それらの中に、新メモリ割り当て量に等しいものが存在しない場合には、その新メモリ割り当て量を超えず、それに最も近いメモリ所要量を選択し、それに対応するパラメータファイルを、再起動に用いるパラメータファイルとして選択する（Ｓ１１４５）。

【０１１１】監視エージェント１０は選択した再起動用パラメータファイルを用いてＤＢシステムの再起動を行う（Ｓ１１５０）。そして、サーバ側メッセージ交換モジュールに対してＤＢ起動済み信号を送信する（Ｓ１１５５）。

【０１１２】サーバ側メッセージ交換モジュールはＤＢ起動済み信号を受信に応じたセッション処理（後述する）を行う（Ｓ１１６０）。以上のようにして本実施の形態のシステムはメモリ再編成処理を行う。

【０１１３】次に、サーバ側メッセージ交換モジュールにおけるセッション処理について説明する。サーバ側メッセージ交換モジュールは、「セッションＩＤ」、「トランザクションＩＤ」、「端末情報」、「メッセージチェイン」といった情報の組をセッション毎に格納したセッション情報テーブルを有している。

【０１１４】ここで、「セッションＩＤ」は、セッション毎に一意に割り当てられる識別子である。これはＤＢプロセスによって割り当てられ、サーバ側メッセージ交換モジュールは、ＤＢプロセスからセッションＩＤを通知される。

【０１１５】「トランザクションＩＤ」はトランザクションを管理するための識別番号であり、これは、ＤＢプロセスによって割り当てられ、サーバ側メッセージ交換モジュールは、トランザクションが発生した時点でＤＢプロセスからトランザクションＩＤを通知される。なお、「トランザクション」とは、あるセッションにおいて、ＤＢに対して変更、追加、削除の操作が一回以上行われる場合、この操作が最初に行われた時点から“確定”メッセージがＤＢプロセスによって受け付けられるまでをいう。

【０１１６】「端末情報」は端末と通信するためにネットワーク制御層において必要とされる情報である。ちなみに、これはどのような種類のネットワーク（例えばＴＣＰ／ＩＰ等）を使用するかに応じて異なり得る。

【０１１７】「メッセージチェイン」は、ＤＢシステム動作ノード１０１０がエンドユーザ側ノード１０００から受け取ったメッセージのうち、その時点ではＤＢプロセスに渡すことができないものを保留するための仕組みである。図１６は、メッセージチェインの構造を説明する模式図である。セッション情報テーブル中には、メッセージチェインの先頭アドレスを表すメッセージチェインポインタ１２００が格納される。メッセージチェインポインタ１２００が指し示すメモリ上、又は磁気ディスク上のアドレスから連続する領域に先頭のメッセージエントリ１２０２が格納され、例えば、そのメッセージエントリの前部にはメッセージの実体が、また後部には次のメッセージエントリの先頭アドレスを示す次のポインタ１２０４が格納される。このように各メッセージエントリがそれに続くメッセージエントリの開始アドレス情報を有することにより各メッセージエントリは順に接続された鎖状の構造を形成する。ちなみに、サーバ側メッセージ交換モジュールは、先頭のメッセージについてＤＢプロセスに渡す等の処理を済ますと、その先頭メッセージのエントリに格納された次メッセージエントリへのポインタ１２０４をメッセージチェインポインタ１２００に転記して、当該先頭メッセージエントリをチェインから外す。

【０１１８】サーバ側メッセージ交換モジュールには、以下に説明する「通常状態」、「準備中状態」、「準備済み状態」の3つの状態が定義される。

【０１１９】「通常状態」は、サーバ側メッセージ交換モジュールが再起動要求信号を受け取っていない状態で、エンドユーザアプリケーションとＤＢプロセスとが制約なしにメッセージ交換できる状態である。

【０１２０】「準備中状態」は、再起動要求信号を受信し、一つ以上のセッション情報にトランザクションＩＤが登録されている状態であり、エンドユーザアプリケーションとＤＢプロセスとのメッセージ交換には次の制約が課される。

【０１２１】（１）新規のセッション要求は受け付けない、（２）セッション情報にトランザクションＩＤが設定されているセッションは当該トランザクションが確定、終了されるまでメッセージ交換を可能とする、（３）セッション情報にトランザクションＩＤが設定されていないセッションについてエンドユーザアプリケーションからメッセージが届いたとき（例えば、前トランザクションが終了したセッションについて新たなトランザクションの開始となるメッセージが届いたとき）は、

25

当該メッセージをメッセージチェインに登録しDBプロセスに伝達しない。

【0122】すなわち、準備中状態では、新たなセッション、トランザクションの開設は認めず、その一方で現存のトランザクションについてのみDBに対する実際の処理を認め、その正常な終了を待つわけである。

【0123】「準備済み状態」は、再起動要求信号を受信し、セッション情報が全く無い状態、または全てのセッション情報にトランザクションIDが登録されていない状態であり、この場合には、エンドユーザアプリケーションとDBプロセスとのメッセージ交換には次の制約が課される。

【0124】(1)新規のセッション要求は受け付けない、(2)セッション情報にエンドユーザアプリケーションからメッセージが届いたときは、当該メッセージをメッセージチェインに登録しDBプロセスに伝達しない。

【0125】すなわち、準備済み状態は、DBシステムを停止できる状態であり、メモリ再編成処理のための準備ができたという意味である。この状態は、監視エージェント10からのDB起動済み信号によって「通常状態」へ遷移する。

【0126】図17～20は、サーバ側メッセージ交換モジュールでの処理を説明するためのフロー図である。これらの図は、便宜上一つのフロー図を分割したものであり、それらに現れる同一記号で示されるノード同士は互いに接続されることを意味する。例えば、図17の円内に"F"を記したノードAから、図18の同様に表されるノードFへ処理が渡されることになる。

【0127】サーバ側メッセージ交換モジュールは、DBプロセスからのメッセージ、ネットワークを介したエンドユーザアプリケーションからのメッセージ、監視エージェント10からのメッセージを受け取ると(S1300)、それが監視エージェント10からのメッセージであるか否かを判別する(S1305)。判定結果が"Yes"である場合には、次に、当該メッセージが再起動要求信号であるかどうかが判定される(S1310)。判定結果が"Yes"であり、さらに続く判定によりセッション情報が存在し(S1315)、かつ既存セッション情報を調査した結果、トランザクションIDが設定されている場合には(S1320)、サーバ側メッセージ交換モジュールの状態を準備中状態とし(S1325)、ステップS1300に戻る。ステップS1315、S1320において判定結果が"No"である場合には、稼働中のトランザクションは既に存在していないので、サーバ側メッセージ交換モジュールの状態を準備済み状態として(S1330)ステップS1300に戻る。

【0128】ステップS1310において監視エージェント10からのメッセージが再起動要求信号でない場合

26

は、DB起動済み信号であると判断される。この場合には、セッション情報のメッセージチェインに保留されているメッセージが存在するならば(S1335)、全てのセッション情報において保留されているメッセージをDBプロセスに送信してから(S1340)、一方、メッセージが存在しないならばそのまま直ちに、サーバ側メッセージ交換モジュールの状態を通常状態に変更する(S1345)。

【0129】また、ステップS1300で受け取ったメッセージがDBプロセスからのものである場合には(S1305，S1400)、それがトランザクション確定済みメッセージであるかどうかが判定される(S1405)。判定結果が"Yes"である場合には、当該メッセージがどのセッションに対する返答メッセージかを調べ、そのセッションに対するセッション情報のトランザクションIDをクリアする(S1410)。その結果、全てのセッションのトランザクションIDがクリアされた場合(S1415)、サーバ側メッセージ交換モジュールの状態が準備中であれば(S1420)、それを準備済み状態に変更し、監視エージェント10に準備済み信号を送信して(S1425)、ステップS1300に戻る。一方、まだトランザクションIDが確定していないセッションがある場合は(S1415)、サーバ側メッセージ交換モジュールはそれまでの状態(通常状態又は準備中状態)を維持しステップS1300に戻る。また、ステップS1420においてサーバ側メッセージ交換モジュールの状態が準備中でない場合、具体的には通常状態である場合にも、サーバ側メッセージ交換モジュールはそれまでの通常状態を維持して、ステップS1300に戻る。なお、ステップS1420では、先行するステップS1410までにおいてまだトランザクションが存在したので準備済み状態ではあり得ない。

【0130】ステップS1405において、DBプロセスからのメッセージがトランザクション確定済みメッセージではなく(S1405)、かつ当該メッセージ中にトランザクションIDが含まれている場合(S1430)には、新たなトランザクションの発生を示す返答メッセージである。この場合には、当該メッセージがどのセッションに対する返答メッセージであるかを調べ、そのセッションに対応するセッション情報に当該トランザクションIDを設定し(S1435)、セッション情報に基づいてエンドユーザ側ノード1000を特定し当該メッセージを返し(S1440)、ステップS1300に戻る。なお、サーバ側メッセージ交換モジュールの状態が準備中である場合には、このような場合は生じない。なぜなら、新たなトランザクションはエンドユーザアプリケーションからの要求に応じて発生するものであるが、サーバ側メッセージ交換モジュールの状態が準備中である場合には、後述するようにエンドユーザアプリケーションからのメッセージはメッセージチェインに格

納されＤＢプロセスには伝達されないので、トランザクションが発生されることも、それを通知するメッセージが生じることもないからである。

【０１３１】一方、ステップＳ１４３０において、判定結果が“Ｎｏ”である場合には、現存するトランザクションに対する処理に応じたメッセージであるので、セッション情報にトランザクションＩＤを設定せずに、ステップＳ１４４０の処理を行って、ステップＳ１３００に戻る。

【０１３２】また、ステップＳ１３００で受け取ったメッセージがエンドユーザ側ノード１０００からのものである場合には（Ｓ１３０５，Ｓ１４００）、処理はサーバ側メッセージ交換モジュールの状態が通常状態であるか否かで分岐する（Ｓ１５００）。

【０１３３】通常状態である場合には（Ｓ１５００）、図１９に示す処理が行われる。まず、当該メッセージが新規セッション確立要求である場合には（Ｓ１５０５）、ＤＢプロセスにメッセージを送信する（Ｓ１５１０）。そして、それに応じてＤＢプロセスがセッションを確立する処理を行い（Ｓ１５１５）、セッション確立通知をサーバ側メッセージ交換モジュールに返すと、サーバ側メッセージ交換モジュールはそのＤＢプロセスからの応答からセッションＩＤを取得し、初期化状態の新規のセッション情報を登録し（Ｓ１５２０）、ステップＳ１３００に戻る。

【０１３４】一方、ステップＳ１５０５にて、新規セッション確立要求ではないと判定された場合には、ＤＢプロセスにメッセージを送信し（Ｓ１５２５）、ステップＳ１３００に戻る。

【０１３５】ステップＳ１５００において通常状態ではないと判定された場合、具体的にはサーバ側メッセージ交換モジュールが準備中又は準備済みの状態である場合には、上述したようなエンドユーザ側ノード１０００とＤＢプロセスとのメッセージ交換には制約が課せられる。まずメッセージが新規セッション確立要求である場合（Ｓ１６００）には、エンドユーザ側ノード１０００にエラーが返され（Ｓ１６０５）、ステップＳ１３００に戻る。

【０１３６】メッセージが新規セッション確立要求でない場合（Ｓ１６００）には、サーバ側メッセージ交換モジュールの状態が準備済み状態か、準備中状態かに応じて分岐する（Ｓ１６１０）。準備済み状態の場合には、セッション情報のメッセージチェインに当該メッセージを登録して（Ｓ１６１５）ステップＳ１３００に戻る。準備中状態の場合には、メッセージからセッションを特定する（Ｓ１６２０）。そして、対応するセッション情報にトランザクションＩＤが登録されている場合には、ＤＢプロセスにメッセージを伝達し（Ｓ１６３０）、ステップＳ１３００に戻る。トランザクションＩＤが登録されていない場合には、メッセージチェインへの登録処理Ｓ１６１５を行い、ステップＳ１３００に復帰する。

図２１は、セッション情報にトランザクションＩＤが登録されているセッションに生じる状態遷移の例を示す状態遷移図である。図に於いて、時間は上から下に経過し、各状態間の実線の矢印は、エンドユーザ側ノード１０００、サーバ側メッセージ交換モジュール及びＤＢプロセス間のメッセージ等の送受信を表し、サーバ側メッセージ交換モジュールの通常状態、準備中状態、及び準備済み状態間の点線の矢印はそれら状態間の状態遷移を表す。

【０１３７】エンドユーザ側ノード１０００とＤＢプロセスとは、通常状態のサーバ側メッセージ交換モジュールを介してメッセージ送信、それに対する結果通知を行っている（Ｐ１７００）。この状態でサーバ側メッセージ交換モジュールは、監視エージェント１０からフェールオーバのためのＤＢ再起動要求信号を受信すると、準備中状態に遷移する（Ｓ１７０５）。以降、エンドユーザ側ノード１０００とＤＢプロセスとは、既存のトランザクションがあるうちは、準備中状態のサーバ側メッセージ交換モジュールを介して、メッセージ送信、それに対する結果通知を行って処理を進める（Ｐ１７１０）。そして、エンドユーザ側ノード１０００からＤＢプロセスにトランザクションが確定したことが通知され、それに対する確定処理済みの通知がＤＢプロセスからサーバ側メッセージ交換モジュールに送信されると（Ｓ１７２０）、サーバ側メッセージ交換モジュールは状態を準備済み状態に遷移させ（Ｓ１７２５）、監視エージェント１０に対し、ＤＢ再起動処理の準備ができたことを通知する準備済み信号を発信する。以降、監視エージェント１０はＤＢ再起動処理を実施し（Ｓ１７３０）、それを終了するとサーバ側メッセージ交換モジュールに対しＤＢ起動済み信号を送信する。サーバ側メッセージ交換モジュールは、この準備済み信号送信からＤＢ起動済み信号受信までの間、エンドユーザ側ノード１０００からメッセージＳ１７３５を受信しても、それをＤＢプロセスに渡さずメッセージチェインに保留する。そしてサーバ側メッセージ交換モジュールは、ＤＢ起動済み信号を受信すると、状態を準備済みから通常状態に復帰させる（Ｓ１７４０）とともに、メッセージチェインに保留されていたメッセージをＤＢプロセスに渡す（Ｓ１７４５）。ＤＢプロセスの保留メッセージに対する処理は、保留されずにそのまま渡されたメッセージに対するものと変わりなく、ＤＢプロセスはその処理結果をエンドユーザ側ノード１０００に通知する（Ｓ１７５０）。

【０１３８】以上のような本実施の形態の仕組みによれば、フェールオーバ対象のＤＢシステム起動に必要な空きメモリ量を確保するためのメモリ再編成処理において稼働中のＤＢシステムを停止する場合に、実行中のトランザクションは正常に終了され、またエンドユーザが発したメッセージは拒否されることなく、例えばＤＢ再起

29

動中においてもメッセージチェインに保留され、再起動終了後に引き続いて対応する処理が行われる。つまり、DB再起動時におけるDBプロセスの処理内容自体の継続性が担保されるとともに、エンドユーザに対するインターフェースとしても、DBシステムが途切れることなく処理を行っているように見せることができる。

【0139】

【発明の効果】本発明に係る計算機システムによれば、ネットワーク接続された各ノードがそれぞれの稼働状態を監視し、その監視結果に基づいて実行障害を生じたノードの処理を引き継がせる代替ノードを選択するので、代替ノードとしてそのときのシステムの状態に応じた最適なものが動的に選択されるという効果が得られる。

【0140】また本発明に係る計算機システムによれば、各ノードのプロセッサ負荷の監視結果に基づいて代替ノードが選択される。例えばプロセッサ負荷が最小になるノードが代替ノードに選択される。これにより、プロセッサ負荷が厳しくないノードが代替ノードに選択され、プロセッサ負荷の面でシステム内での負荷分散が図られ、処理の速度の確保が図られるという効果が得られる。

【0141】また本発明に係る計算機システムによれば、各ノードのメモリ空き容量の監視結果に基づいて代替ノードが選択される。例えばメモリ空き容量が最大になるノードが代替ノードに選択される。これにより、メモリの余裕の比較的大きいノードが代替ノードに選択され、メモリ容量の面でシステム内での負荷分散が図られ、スワッピングが抑制されることによりキャッシュのヒット率が向上したり、スラッシングが防止されることにより高い処理効率が得られるといった効果が得られる。

【0142】また本発明に係る計算機システムによれば、各ノードのプロセッサ負荷とメモリ空き容量との双方の監視結果に基づいて代替ノードが選択される。これにより、プロセッサ負荷及びメモリ容量の両方の面を考慮してシステム内での負荷分散が図られ、処理効率が確保されるという効果が得られる。

【0143】また本発明に係る計算機システムによれば、代替ノードのメモリ空き容量が実行障害を生じた処理プログラムの引き継ぎに不足している場合には、代替ノードで先行して起動されているプログラムへの割り当てメモリ容量が縮小されるので、フェールオーバ処理が一層、確実に実行されるという効果がある。

【0144】また本発明に係る計算機システムによれば、先起動プログラムを停止してメモリ容量の割当変更を行う際に、セッション情報に基づいて既設セッション中のトランザクションが継続中か否かを判別して継続中のトランザクションに対するメッセージのみが先起動プログラムに渡される。また当該セッションに対する他のメッセージはメモリ容量の割当変更が完了するまでセッション情報に保留され、割当変更後に通常通りに処理される。これらにより、先起動プログラムを停止し、そのメモリ使用量を縮小してフェールオーバ対象の処理プログラム起動に必要な空きメモリ量を確保する場合に、先起動プログラムの処理内容の継続性が担保され、またエンドユーザには先起動プログラムが停止されることが意識されないインターフェースが提供されるという効果が得られる。

30

【図面の簡単な説明】

【図1】　本発明の実施の形態に係る計算機システムの構成を説明するための模式図である。

【図2】　本発明の実施の形態に係る監視情報リポジトリに格納される情報の一例の論理構成を示す模式図である。

【図3】　本発明の実施の形態に係る監視エージェントのDBアプリケーション状態監視に関する初期処理を説明するフロー図である。

【図4】　本発明の実施の形態に係る監視エージェントの監視処理の概略を説明するフロー図である。

【図5】　本発明の実施の形態に係る監視エージェントによる自ノードに対するDB状況チェック処理S125を説明するフロー図である。

【図6】　本発明の実施の形態に係る監視エージェントによる全ノード処理を説明するフロー図である。

【図7】　本発明の実施の形態に係る監視エージェントによる全ノード処理を説明するフロー図である。

【図8】　実施の形態2に係る監視エージェントによるDBアプリケーションダウン時の処理を示すフロー図である。

【図9】　実施の形態2に係る監視エージェントによるノードダウン時の処理を示すフロー図である。

【図10】　実施の形態2に係る監視エージェントにおけるフェールオーバ指示に応じて行われるDBアプリケーション再起動処理を説明するフロー図である。

【図11】　実施の形態2に係る監視エージェントによるDBアプリケーションの実行障害検出時の動作の一例を説明する模式図である。

【図12】　実施の形態3に係る監視エージェントの代替ノード選択方法を説明するフロー図である。

【図13】　実施の形態5に係る監視エージェントによる、メモリ再編成を含んだフェールオーバ処理を説明するフロー図である。

【図14】　実施の形態6に係る計算機システムにおけるエンドユーザ側ノードとDBシステム動作ノードとの機能レイヤ構成例を示す模式図である。

【図15】　実施の形態6に係る監視エージェントによるメモリ再編成処理を説明するフロー図である。

【図16】　実施の形態6に係るメッセージチェインの構造を説明する模式図である。

【図17】　実施の形態6に係るサーバ側メッセージ交

31

換モジュールでの処理を説明するためのフロー図である。

【図18】　実施の形態6に係るサーバ側メッセージ交換モジュールでの処理を説明するためのフロー図である。

【図19】　実施の形態6に係るサーバ側メッセージ交換モジュールでの処理を説明するためのフロー図である。

【図20】　実施の形態6に係るサーバ側メッセージ交換モジュールでの処理を説明するためのフロー図である。

【図21】　実施の形態6における、セッション情報に

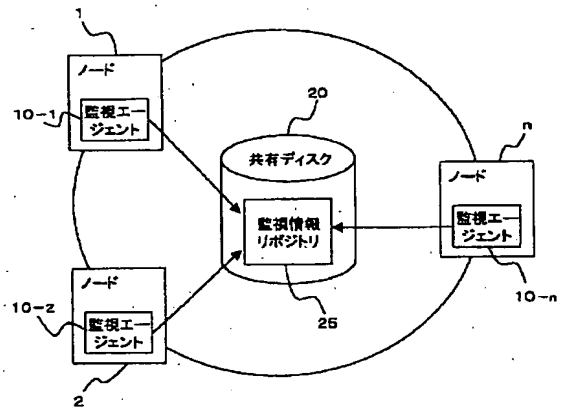
32

トランザクションIDが登録されているセッションに生じる状態遷移の例を示す状態遷移図である。

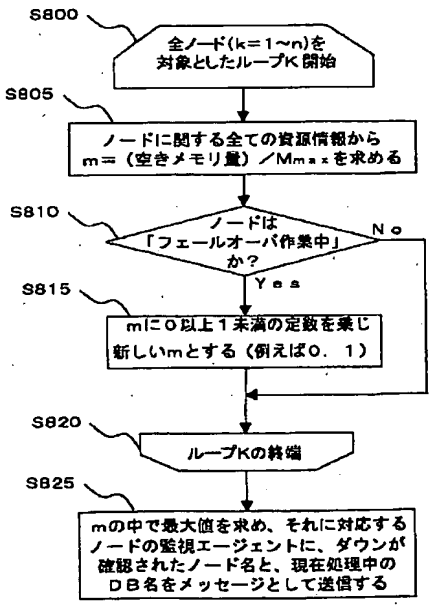
【符号の説明】

1，2，3　ノード、10　監視エージェント、20　共有ディスク、25監視情報リポジトリ、1000　エンドユーザ側ノード、1002　エンドユーザアプリケーション、1004　端末側メッセージ交換モジュール、1006ネットワーク制御層、1010　DBシステム動作ノード、1012　DBアプリケーション、1014　サーバ側メッセージ交換モジュール、1016　ネットワーク制御層、1020　コネクション。

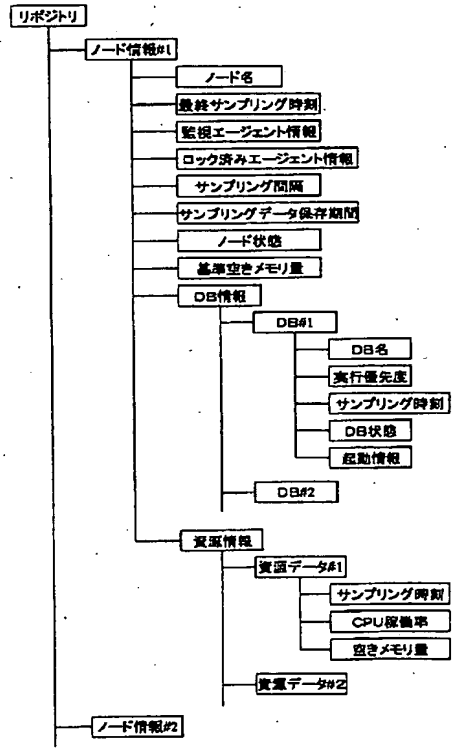
【図1】



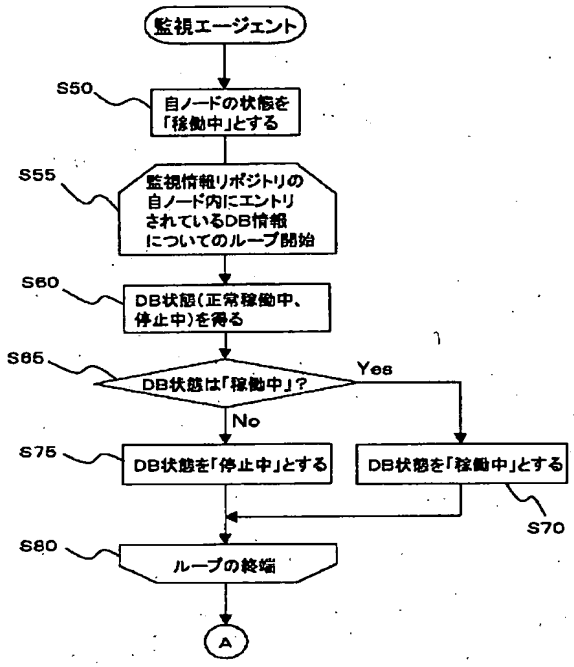
【図12】



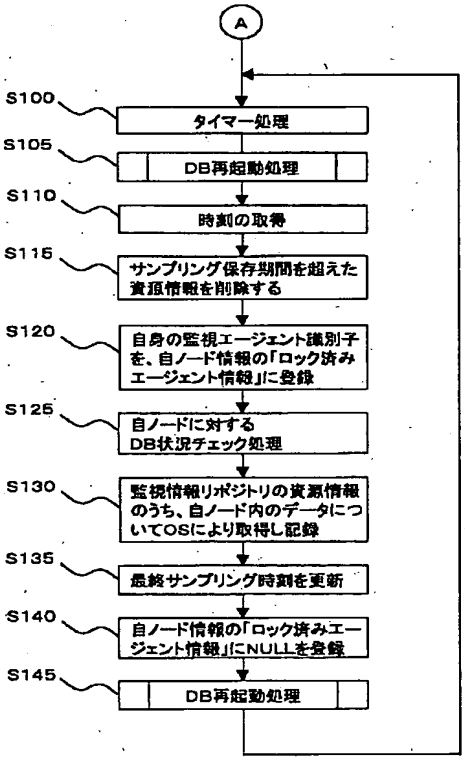
【図2】



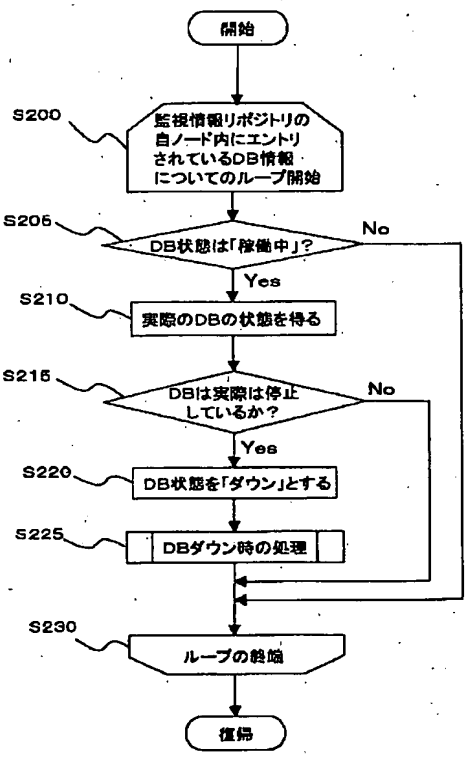
【図3】


監視エージェント
S50
自ノードの状態を「稼働中」とする
S55
監視情報リポジトリの自ノード内にエントリされているDB情報についてのループ開始
S60
DB状態(正常稼働中、停止中)を得る
S65
DB状態は「稼働中」?
Yes
No
S75
DB状態を「停止中」とする
DB状態を「稼働中」とする
S70
S80
ループの終端
A

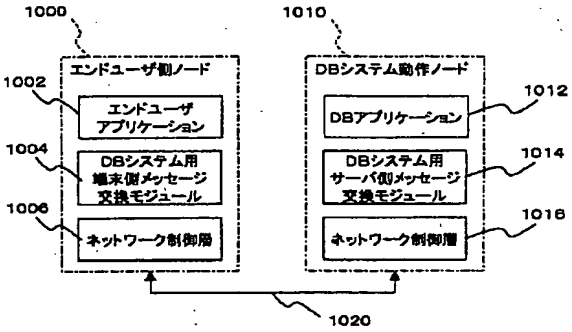
【図4】


A
S100
タイマー処理
S105
DB再起動処理
S110
時刻の取得
S115
サンプリング保存期間を超えた資源情報を削除する
S120
自身の監視エージェント識別子を、自ノード情報の「ロック済みエージェント情報」に登録
S125
自ノードに対するDB状況チェック処理
S130
監視情報リポジトリの資源情報のうち、自ノード内のデータについてOSにより取得し記録
S135
最終サンプリング時刻を更新
S140
自ノード情報の「ロック済みエージェント情報」にNULLを登録
S145
DB再起動処理

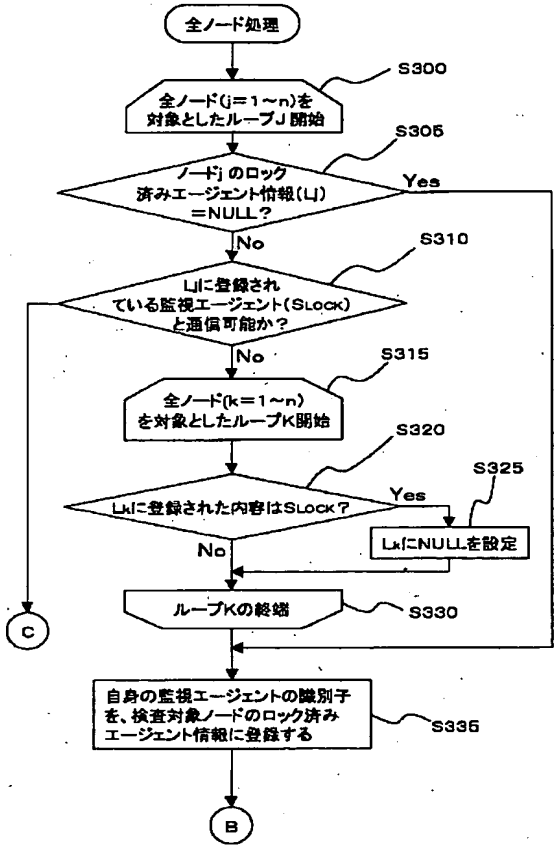
【図5】


開始
S200
監視情報リポジトリの自ノード内にエントリされているDB情報についてのループ開始
S205
DB状態は「稼働中」?
No
Yes
S210
実際のDBの状態を得る
S215
DBは実際は停止しているか?
No
Yes
S220
DB状態を「ダウン」とする
S225
DBダウン時の処理
S230
ループの終端
復帰

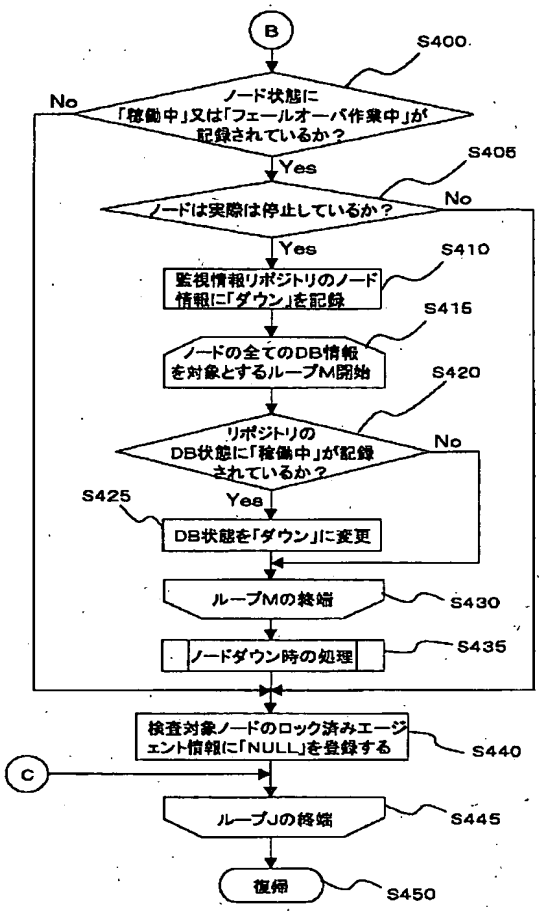
【図14】


1000
エンドユーザ側ノード
1002
エンドユーザアプリケーション
1004
DBシステム用端末側メッセージ交換モジュール
1006
ネットワーク制御層
1010
DBシステム動作ノード
1012
DBアプリケーション
1014
DBシステム用サーバ側メッセージ交換モジュール
1016
ネットワーク制御層
1020

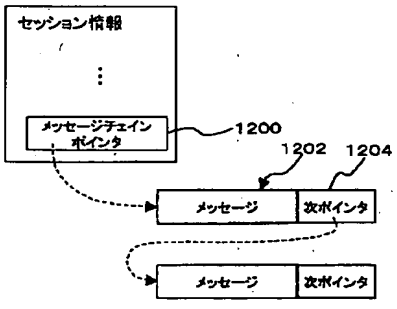
【図６】


全ノード処理
S300
全ノード(j=1～n)を対象としたループJ開始
S305
ノードjのロック済みエージェント情報(Lj)=NULL?
Yes
No
S310
Ljに登録されている監視エージェント(SLock)と通信可能か?
No
S315
全ノード(k=1～n)を対象としたループK開始
S320
Lkに登録された内容はSLock?
Yes
S325
LkにNULLを設定
No
ループKの終端
S330
C
自身の監視エージェントの識別子を、検査対象ノードのロック済みエージェント情報に登録する
S335
B

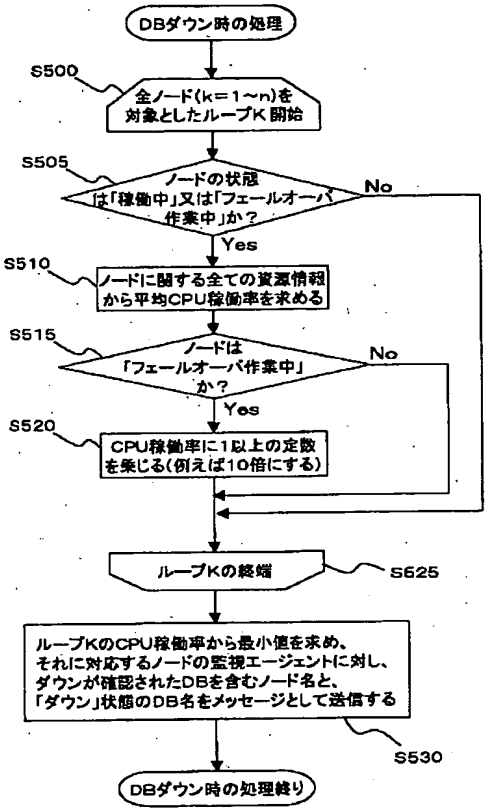
【図７】


B
S400
ノード状態に「稼働中」又は「フェールオーバ作業中」が記録されているか?
No
Yes
S405
ノードは実際は停止しているか?
No
Yes
S410
監視情報リポジトリのノード情報に「ダウン」を記録
S415
ノードの全てのDB情報を対象とするループM開始
S420
リポジトリのDB状態に「稼働中」が記録されているか?
No
S425
Yes
DB状態を「ダウン」に変更
ループMの終端
S430
ノードダウン時の処理
S435
検査対象ノードのロック済みエージェント情報に「NULL」を登録する
S440
C
ループJの終端
S445
復帰
S450

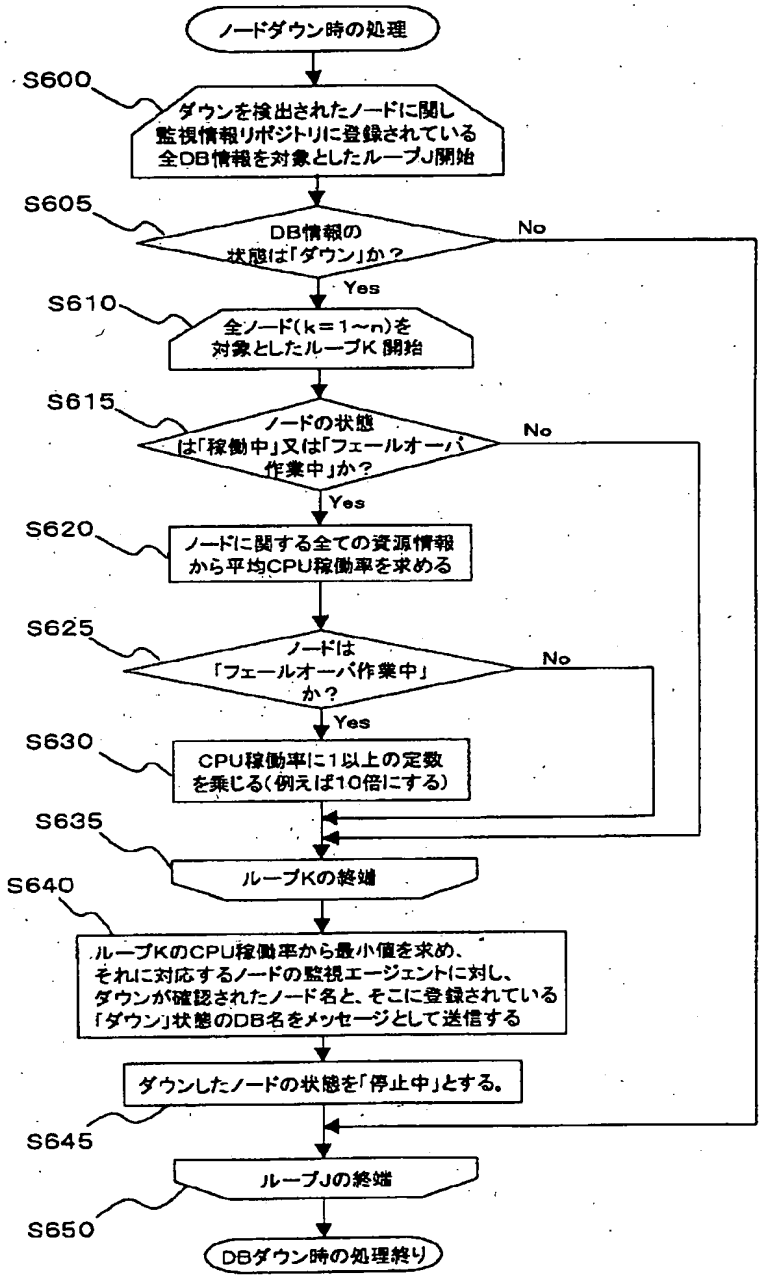
【図１６】


セッション情報
メッセージチェインポインタ
1200
1202
1204
メッセージ
次ポインタ
メッセージ
次ポインタ

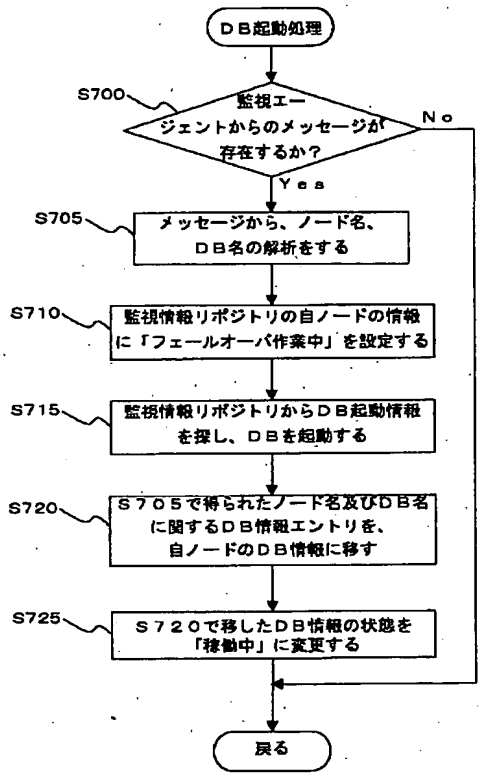
【 図8 】


DBダウン時の処理
S500
全ノード(k＝1～n)を
対象としたループK 開始
S505
ノードの状態
は「稼働中」又は「フェールオーバ
作業中」か？
No
Yes
S510
ノードに関する全ての資源情報
から平均CPU稼働率を求める
S515
ノードは
「フェールオーバ作業中」
か？
No
Yes
S520
CPU稼働率に1以上の定数
を乗じる(例えば10倍にする)
ループKの終端
S625
ループKのCPU稼働率から最小値を求め、
それに対応するノードの監視エージェントに対し、
ダウンが確認されたDBを含むノード名と、
「ダウン」状態のDB名をメッセージとして送信する
S530
DBダウン時の処理終り

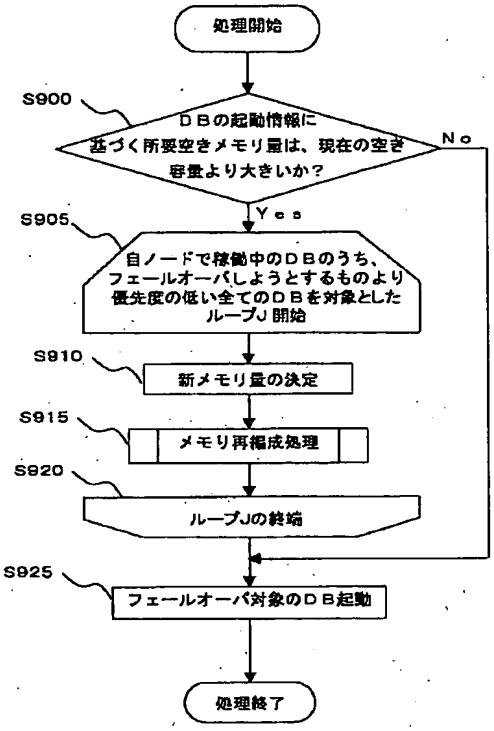
【 図9 】


ノードダウン時の処理
S600
ダウンを検出されたノードに関し
監視情報リポジトリに登録されている
全DB情報を対象としたループJ開始
S605
DB情報の
状態は「ダウン」か？
No
Yes
S610
全ノード(k＝1～n)を
対象としたループK 開始
S615
ノードの状態
は「稼働中」又は「フェールオーバ
作業中」か？
No
Yes
S620
ノードに関する全ての資源情報
から平均CPU稼働率を求める
S625
ノードは
「フェールオーバ作業中」
か？
No
Yes
S630
CPU稼働率に1以上の定数
を乗じる(例えば10倍にする)
S635
ループKの終端
S640
ループKのCPU稼働率から最小値を求め、
それに対応するノードの監視エージェントに対し、
ダウンが確認されたノード名と、そこに登録されている
「ダウン」状態のDB名をメッセージとして送信する
ダウンしたノードの状態を「停止中」とする。
S645
ループJの終端
S650
DBダウン時の処理終り

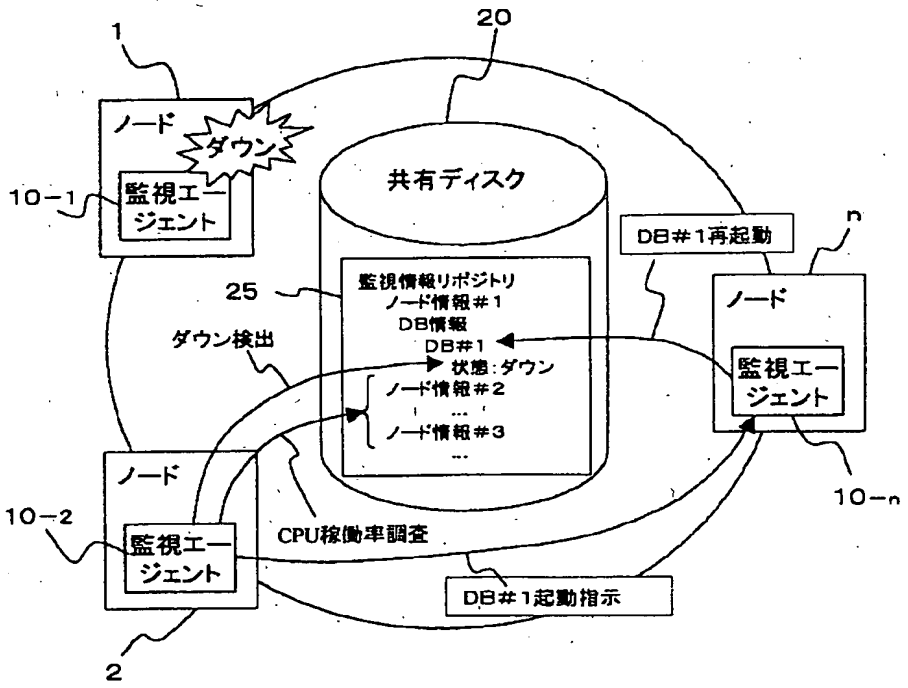
【図１０】


ＤＢ起動処理
S700
監視エージェントからのメッセージが存在するか？
No
Yes
S705
メッセージから、ノード名、ＤＢ名の解析をする
S710
監視情報リポジトリの自ノードの情報に「フェールオーバ作業中」を設定する
S715
監視情報リポジトリからＤＢ起動情報を探し、ＤＢを起動する
S720
Ｓ７０５で得られたノード名及びＤＢ名に関するＤＢ情報エントリを、自ノードのＤＢ情報に移す
S725
Ｓ７２０で移したＤＢ情報の状態を「稼働中」に変更する
戻る

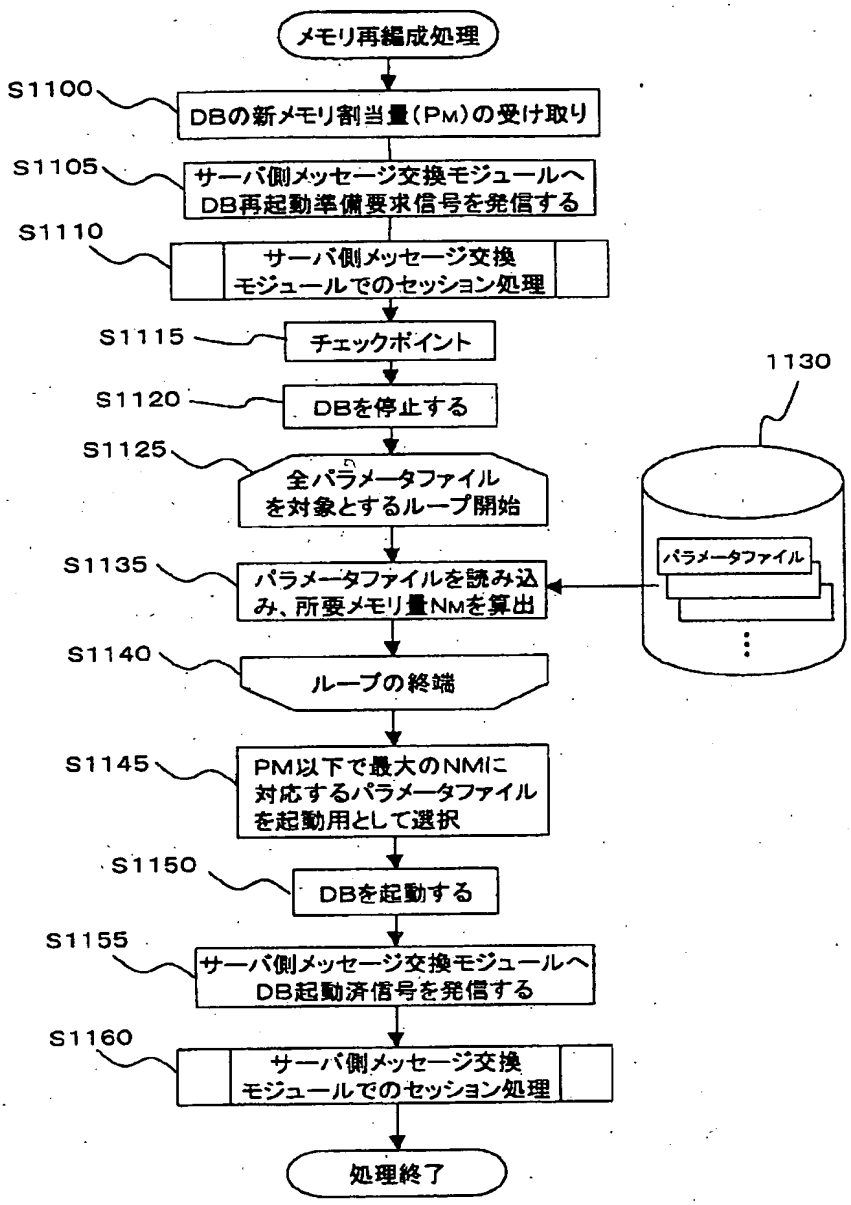
【図１３】


処理開始
S900
ＤＢの起動情報に基づく所要空きメモリ量は、現在の空き容量より大きいか？
No
Yes
S905
自ノードで稼働中のＤＢのうち、フェールオーバしようとするものより優先度の低い全てのＤＢを対象としたループJ 開始
S910
新メモリ量の決定
S915
メモリ再編成処理
S920
ループJの終端
S925
フェールオーバ対象のＤＢ起動
処理終了

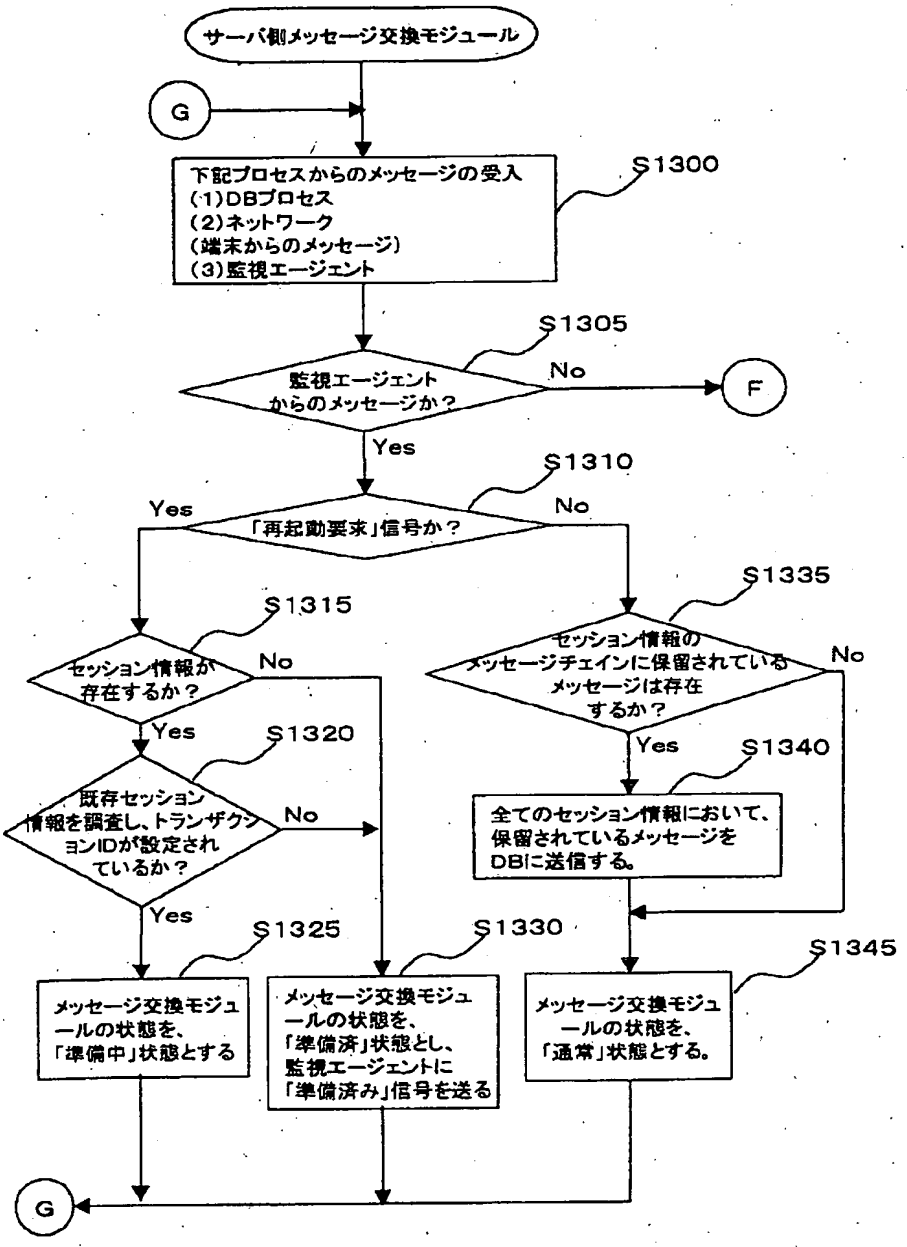
【図１１】


1
20
ノード
ダウン
監視エージェント
10−1
共有ディスク
DB#1再起動
n
25
監視情報リポジトリ
ノード情報#1
DB情報
DB#1
状態:ダウン
ノード情報#2
...
ノード情報#3
...
ダウン検出
ノード
監視エージェント
ノード
10−2
監視エージェント
CPU稼働率調査
10−n
DB#1起動指示
2

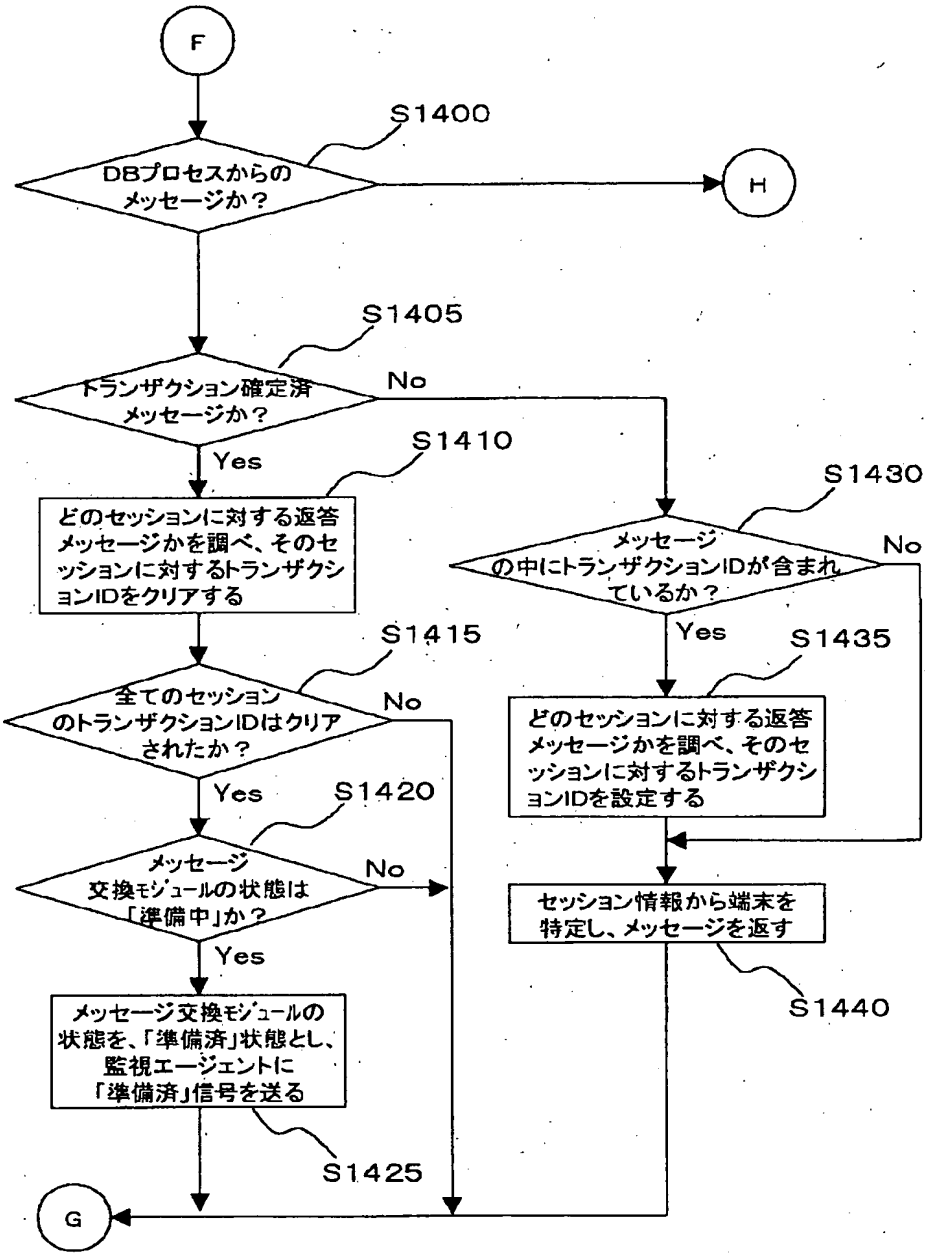
【 図15 】


メモリ再編成処理
S1100
DBの新メモリ割当量(PM)の受け取り
S1105
サーバ側メッセージ交換モジュールへ
DB再起動準備要求信号を発信する
S1110
サーバ側メッセージ交換
モジュールでのセッション処理
S1115
チェックポイント
S1120
DBを停止する
S1125
全パラメータファイル
を対象とするループ開始
S1135
パラメータファイルを読み込
み、所要メモリ量NMを算出
S1140
ループの終端
S1145
PM以下で最大のNMに
対応するパラメータファイル
を起動用として選択
S1150
DBを起動する
S1155
サーバ側メッセージ交換モジュールへ
DB起動済信号を発信する
S1160
サーバ側メッセージ交換
モジュールでのセッション処理
処理終了
1130
パラメータファイル

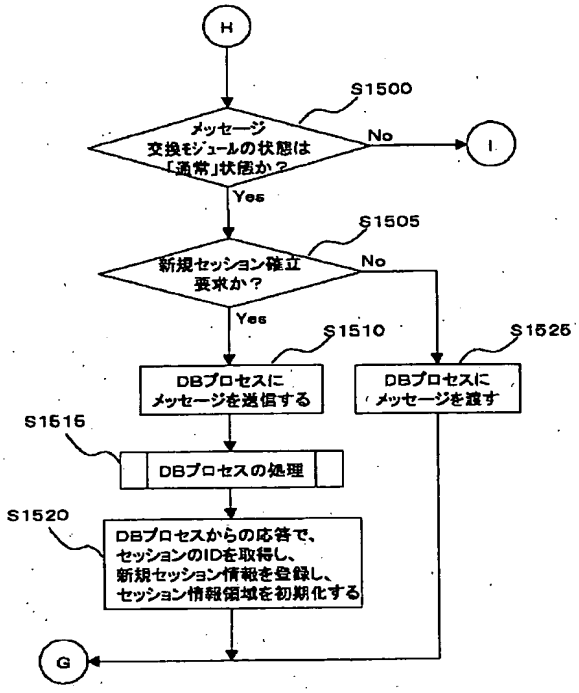
【図１７】


サーバ側メッセージ交換モジュール
G
下記プロセスからのメッセージの受入
(1)DBプロセス
(2)ネットワーク
(端末からのメッセージ)
(3)監視エージェント
S1300
S1305
監視エージェント
からのメッセージか？
No
F
Yes
S1310
Yes
「再起動要求」信号か？
No
S1315
セッション情報が
存在するか？
No
Yes
S1320
既存セッション
情報を調査し、トランザクン
ョンIDが設定され
ているか？
No
Yes
S1335
セッション情報の
メッセージチェインに保留されている
メッセージは存在
するか？
No
Yes
S1340
全てのセッション情報において、
保留されているメッセージを
DBに送信する。
S1325
メッセージ交換モジュ
ールの状態を、
「準備中」状態とする
S1330
メッセージ交換モジュ
ールの状態を、
「準備済」状態とし、
監視エージェントに
「準備済み」信号を送る
S1345
メッセージ交換モジュ
ールの状態を、
「通常」状態とする。
G

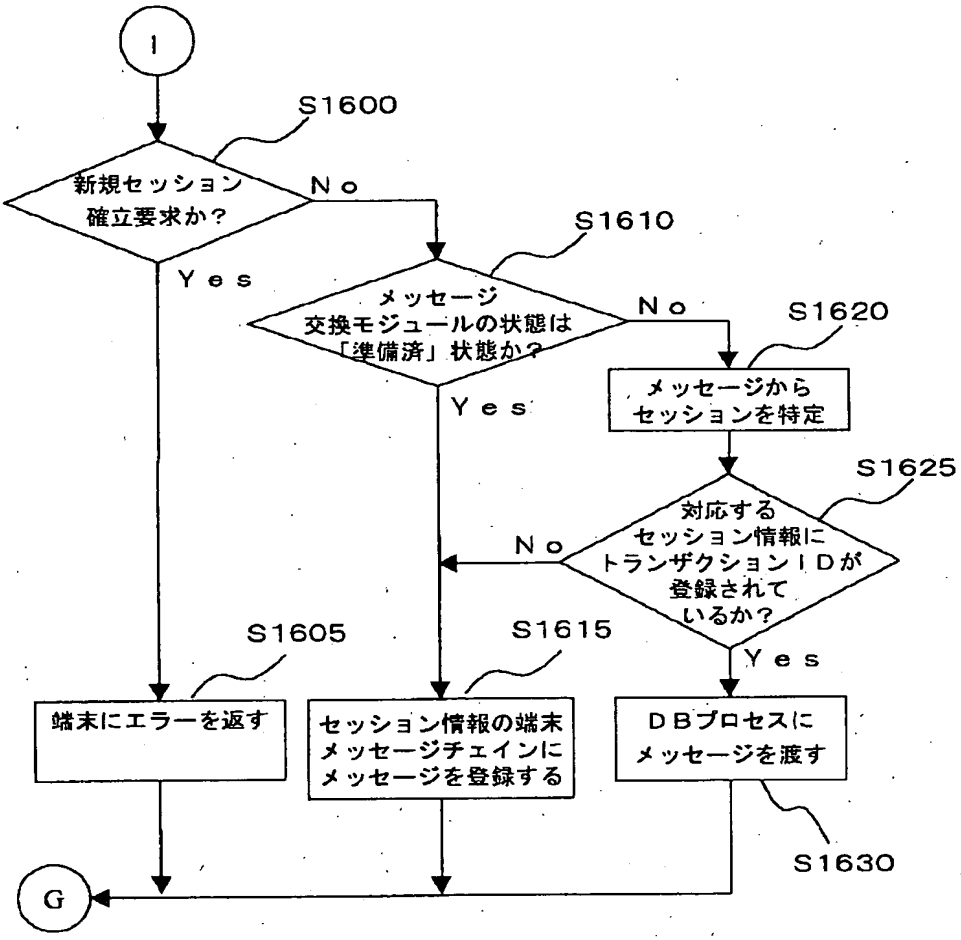
【図18】


F
S1400
DBプロセスからの
メッセージか？
H
S1405
トランザクション確定済
メッセージか？
No
Yes
S1410
どのセッションに対する返答
メッセージかを調べ、そのセ
ッションに対するトランザクシ
ョンIDをクリアする
S1415
全てのセッション
のトランザクションIDはクリア
されたか？
No
Yes
S1420
メッセージ
交換ﾓｼﾞｭｰﾙの状態は
「準備中」か？
No
Yes
メッセージ交換ﾓｼﾞｭｰﾙの
状態を、「準備済」状態とし、
監視エージェントに
「準備済」信号を送る
S1425
G
S1430
メッセージ
の中にトランザクションIDが含まれ
ているか？
No
Yes
S1435
どのセッションに対する返答
メッセージかを調べ、そのセ
ッションに対するトランザクシ
ョンIDを設定する
セッション情報から端末を
特定し、メッセージを返す
S1440

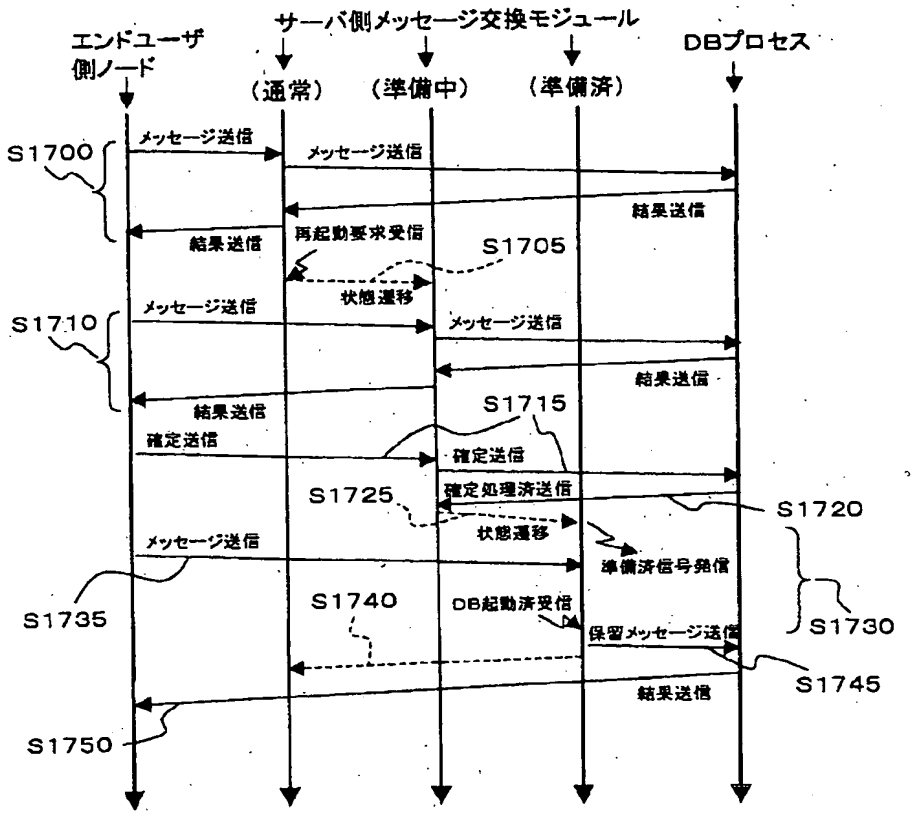
【 図19 】


H
S1500
メッセージ
交換モジュールの状態は
「通常」状態か？
No
I
Yes
S1505
新規セッション確立
要求か？
No
Yes
S1510
DBプロセスに
メッセージを送信する
S1525
DBプロセスに
メッセージを渡す
S1515
DBプロセスの処理
S1520
DBプロセスからの応答で、
セッションのIDを取得し、
新規セッション情報を登録し、
セッション情報領域を初期化する
G

【図２０】


I
S1600
新規セッション
確立要求か？
No
Yes
S1610
メッセージ
交換モジュールの状態は
「準備済」状態か？
No
Yes
S1620
メッセージから
セッションを特定
S1625
対応する
セッション情報に
トランザクションＩＤが
登録されて
いるか？
No
Yes
S1605
端末にエラーを返す
S1615
セッション情報の端末
メッセージチェインに
メッセージを登録する
ＤＢプロセスに
メッセージを渡す
S1630
G

【図２１】



【手続補正書】
【提出日】平成１１年8月１７日（１９９９．８．１７）
【手続補正1】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正内容】
【特許請求の範囲】
【請求項1】　それぞれ処理プログラムを実行可能な複数のノードがネットワーク接続された計算機システムにおいて、
前記各ノードは、
前記各ノードそれぞれの稼働状況を監視する監視手段と、
前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき、代替ノードを選択する代替ノード選択手段と、
前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段と、
を有し、
前記監視手段は、前記各ノードそれぞれのプロセッサ負荷を監視し、
前記代替ノード選択手段は、前記監視手段により得られた前記プロセッサ負荷に基づいて前記代替ノードを選択すること、
を特徴とする計算機システム。
【請求項2】　請求項1記載の計算機システムにおいて、
前記代替ノード選択手段は、前記プロセッサ負荷が最小であるノードを前記代替ノードとして選択すること、
を特徴とする計算機システム。
【請求項3】　それぞれ処理プログラムを実行可能な複数のノードがネットワーク接続された計算機システムにおいて、
前記各ノードは、
前記各ノードそれぞれの稼働状況を監視する監視手段と、
前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき、代替ノードを選択する代

替ノード選択手段と、
前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段と、
を有し、
前記監視手段は、前記各ノードそれぞれのメモリ空き容量を監視し、
前記代替ノード選択手段は、前記監視手段により得られた前記メモリ空き容量に基づいて前記代替ノードを選択すること、
を特徴とする計算機システム。
【請求項4】　請求項3記載の計算機システムにおいて、
前記代替ノード選択手段は、前記メモリ空き容量が最大であるノードを前記代替ノードとして選択すること、
を特徴とする計算機システム。
【請求項5】　それぞれ処理プログラムを実行可能な複数のノードがネットワーク接続された計算機システムにおいて、
前記各ノードは、
前記各ノードそれぞれの稼働状況を監視する監視手段と、
前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき、代替ノードを選択する代替ノード選択手段と、
前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段と、
を有し、
前記監視手段は、前記各ノードそれぞれのプロセッサ負荷とメモリ空き容量とを監視し、
前記代替ノード選択手段は、前記監視手段により得られた前記プロセッサ負荷と前記メモリ空き容量とに基づいて前記代替ノードを選択すること、
を特徴とする計算機システム。
【請求項6】　それぞれ処理プログラムを実行可能な複数のノードがネットワーク接続された計算機システムにおいて、
前記各ノードは、
前記各ノードそれぞれの稼働状況を監視する監視手段と、
前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき、代替ノードを選択する代替ノード選択手段と、
前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段と、
を有し、
前記監視手段は、前記各ノードそれぞれのメモリ空き容量を監視し、
前記フェールオーバ手段は、
前記代替ノードの前記メモリ空き容量が前記処理プログラムの引き継ぎに十分か否かを判断する容量比較手段と、
前記メモリ空き容量が不足するときは、前記代替ノードで先行して起動されている先起動プログラムに割り当てられるメモリ容量を縮小するメモリ割当変更手段と、
を有することを特徴とする計算機システム。
【請求項7】　請求項6記載の計算機システムにおいて、
前記代替ノードの前記先起動プログラムと当該先起動プログラムを前記ネットワークを介して利用するユーザノードとの間のセッション情報を含んだセッション情報テーブルを有し、
前記フェールオーバ手段は、前記メモリ容量の割当変更を行う際に、前記セッション情報に基づいて既設セッション中のトランザクションが継続中か否かを判別して継続中のトランザクションに対するメッセージのみ前記先起動プログラムに渡し、当該セッションに対する他のメッセージは前記割当変更が完了するまで前記セッション情報に保留するメッセージ取扱手段を有し、
前記メモリ割当変更手段は、前記トランザクションが終了したときに、前記先起動プログラムの実行を停止してメモリ割当変更を行うこと、
を特徴とする計算機システム。
【手続補正2】
【補正対象書類名】明細書
【補正対象項目名】０００７
【補正方法】変更
【補正内容】
【０００７】
【課題を解決するための手段】本発明に係る、それぞれ処理プログラムを実行可能な複数のノードがネットワーク接続された計算機システムは、前記各ノードが、前記各ノードそれぞれの稼働状況を監視する監視手段と、前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき代替ノードを選択する代替ノード選択手段と、前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段とを有し、前記監視手段は、前記各ノードそれぞれのプロセッサ負荷を監視し、前記代替ノード選択手段は、前記監視手段により得られた前記プロセッサ負荷に基づいて前記代替ノードを選択することを特徴とする。
【手続補正3】
【補正対象書類名】明細書
【補正対象項目名】０００８
【補正方法】変更
【補正内容】
【０００８】本発明の好適な態様は、前記代替ノード選

択手段が前記プロセッサ負荷が最小であるノードを前記代替ノードとして選択するものである。

【手続補正４】
【補正対象書類名】明細書
【補正対象項目名】０００９
【補正方法】変更
【補正内容】
【０００９】別の本発明に係る計算機システムにおいては、前記各ノードが、前記各ノードそれぞれの稼働状況を監視する監視手段と、前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき代替ノードを選択する代替ノード選択手段と、前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段とを有し、前記監視手段は前記各ノードそれぞれのメモリ空き容量を監視し、前記代替ノード選択手段は前記監視手段により得られた前記メモリ空き容量に基づいて前記代替ノードを選択することを特徴とする。本発明の好適な態様は、前記代替ノード選択手段が前記メモリ空き容量が最大であるノードを前記代替ノードとして選択するものである。

【手続補正５】
【補正対象書類名】明細書
【補正対象項目名】００１１
【補正方法】変更
【補正内容】
【００１１】また別の本発明に係る計算機システムにおいては、前記各ノードが、前記各ノードそれぞれの稼働状況を監視する監視手段と、前記監視手段が任意の前記ノードでの前記処理プログラムの実行障害を検知したとき代替ノードを選択する代替ノード選択手段と、前記実行障害を生じたノードから前記代替ノードへ前記処理プログラムの実行を引き継がせるフェールオーバ手段とを有し、前記監視手段は前記各ノードそれぞれのメモリ空き容量を監視し、前記フェールオーバ手段は前記代替ノードの前記メモリ空き容量が前記処理プログラムの引き継ぎに十分か否かを判断する容量比較手段と、前記メモリ空き容量が不足するときは前記代替ノードで先行して起動されている先起動プログラムに割り当てられるメモリ容量を縮小するメモリ割当変更手段とを有することを特徴とする。